Universidad Nacional de Loja

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Ingeniería en Sistemas

Desarrollo de aplicativo web para la gestión de cobro de mensualidades a estudiantes del preuniversitario CENES

Development of a web application for the management of monthly payments to students of CENES pre-university.

Trabajo de Titulación previo a la obtención del título de Ingeniera en Sistemas

AUTORA:

Carla Isabel Troya Capa

DIRECTOR:

Ing. Edwin René Guamán Quinche, Mg.Sc.

Loja – Ecuador

2023

Certificación

Loja, 24 de febrero de 2023

Ing. Edwin René Guamán Quinche Mg.Sc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

C E R T I F I C O:

Que he revisado y orientado todo el proceso de elaboración del Trabajo Titulación denominado: **Desarrollo de aplicativo web para la gestión de cobro de mensualidades a estudiantes del preuniversitario CENES**, previo a la obtención del título de Ingeniera en Sistemas, de la autoría de la estudiante **Carla Isabel Troya Capa**, con **cédula de identidad** Nro.**1750436592**, una vez que el trabajo cumple con todos los requisitos exigidos por la Universidad Nacional de Loja, para el efecto, autorizo la presentación del mismo para su respectiva sustentación y defensa.

Firma:

Firmado electrónicamente por:
EDWIN RENE GUAMAN QUINCHE

Ing. Edwin René Guamán Quinche Mg.Sc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

Autoría

Yo, **Carla Isabel Troya Capa**, declaró ser autora del presente Trabajo de Titulación y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos o acciones legales por el contenido del mismo. Adicionalmente acepto y autorizo a la Universidad Nacional de Loja, la publicación de mi Trabajo de Titulación en el Repositorio Institucional - Biblioteca Virtual de la UNL.

Firma: 

Cédula de identidad: 1750436592

Fecha: 30 de marzo de 2023

Correo electrónico: citroyac@unl.edu.ec

Teléfono: +593 99-204-5250

Carta de autorización por parte de la autora, para consulta, reproducción parcial o total y/o publicación electrónica del texto completo, del Trabajo de Titulación.

Yo, **Carla Isabel Troya Capa**, declaró ser la autora del Trabajo de Titulación denominado: **Desarrollo de aplicativo web para la gestión de cobro de mensualidades a estudiantes del preuniversitario CENES,** como requisito para optar por el título de **Ingeniera en Sistemas,** autorizo al sistema Bibliotecario de la Universidad Nacional de Loja para que, con fines académicos, muestre la producción intelectual de la Universidad, a través de la visibilidad de su contenido en el Repositorio Institucional.

Los usuarios pueden consultar el contenido de este trabajo en el Repositorio Institucional, en las redes de información del país y del exterior, con los cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia del Trabajo de Titulación que realice un tercero. Para constancia de esta autorización, en la ciudad de Loja, a los treinta días del mes de marzo de dos mil veintitrés.

Firma: Firmado electrónicamente por: CARLA ISABEL TROYA CAPA

Autor: Carla Isabel Troya Capa

Cédula de identidad: 1750436592
Dirección: Loja (Menfis Alto)

Correo electrónico: citroyac@unl.edu.ec

Teléfono: +593 99-204-5250

DATOS COMPLEMENTARIOS:

Director del Trabajo de titulación: Ing. Edwin René Guamán Quinche, Mg. Sc.

Dedicatoria

Dedico este trabajo a Dios por la fuerza y valor dados para culminar esta etapa, a mi madre Mariana (†) que ha sido mi motivación diaria para continuar en el diario vivir, a mi padre Héctor, por todo el apoyo brindado, sacrificio y esfuerzo realizado para que culmine mis estudios; este logro es mío y de él; a mis hermanos y amigos por sus consejos y palabras de aliento dedicadas durante toda esta fase.

Finalmente, dedico este trabajo a Olga y Patricia quienes han sido como una segunda madre: motivándome, apoyándome y al pendiente de mi bienestar en esta trayectoria de estudios.

Carla Isabel Troya Capa

Agradecimiento

Agradezco a la Universidad Nacional de Loja, por permitirme realizarme como profesional, y a todos los docentes que a lo largo de estos años han sido un pilar fundamental para llegar a este punto, con sus conocimientos brindados.

De igual manera un profundo y grato agradecimiento al Ing. Edwin René Guamán Quinche por el tiempo y paciencia brindada al dirigir el desarrollo de este Trabajo de Titulación.

Finalmente, a mi familia por la motivación y apoyo incondicional brindando durante esta trayectoria de estudio.

Carla Isabel Troya Capa

Índice de Contenido:

Índice de Tablas:

Índice de Figuras:

Índice de anexos:

1. Título

Desarrollo de aplicativo web para la gestión de cobro de mensualidades a estudiantes del preuniversitario CENES

2. Resumen

En la actualidad la automatización de procesos se ha vuelto importante para las instituciones públicas, sobre todo si estas brindan servicios que implican una cartera de crédito. Al contar con este tipo de servicio muchas empresas no cuentan con un proceso de gestión adecuado, es muy probable que los recursos que tienen no sean suficientes para cobros de pensiones y se han tenido que utilizar métodos convencionales. Este tipo de inconvenientes no es ajeno a las empresas lojanas de educación como es el caso del Preuniversitario "CENES". En ese contexto se desarrolló un aplicativo web para la gestión de cobros de mensualidades a estudiantes del preuniversitario, con el objetivo de reducir los problemas de inconsistencia de información y de responder a la siguiente pregunta de investigación: ¿El desarrollo de un aplicativo web para la gestión de cobranzas de mensualidades en el preuniversitario CENES, permitirá reducir la tasa de incumplimiento de pagos por parte de los estudiantes?

En tal sentido el marco metodológico utilizado constó de 3 fases. En la primera fase se realizó la documentación de los requerimientos aplicando el estándar IEEE-830. En la fase dos, se diseñó la arquitectura de software utilizando modelos arquitectónicos como Cliente-Servidor y Rest, posteriormente se codificó la solución utilizando Django ResFramework, Angular y flutter. Finalmente, se realizó pruebas de caja negra y blanca, las que permitieron encontrar errores de codificación antes de que el software pase a producción, por otra parte, las pruebas de carga y estrés ayudaron a determinar que los tiempos de respuesta son los adecuados para la petición entre el cliente y el servidor.

Por último, el aplicativo web se ejecutó en un ambiente controlado, evidenciando que el funcionamiento influye significativamente en la gestión y proceso de cobros en el preuniversitario CENES.

Palabras Claves: Desarrollo de Software, gestión de cobros, metodología ICONIX. Django ResFramework

2.1. Abstract

Currently, automation of processes have become extremely important for companies or public institutions, especially if they provide services that involve a credit portfolio. By having this type of service, many companies do not have an adequate management process since it is very likely that the resources they have are not enough to collect pensions or tuition fees and conventional methods have had to be used. This type of inconvenience is not alien to local education companies, as is the case of the "CENES" Pre-University. In this context, a web application was developed for the management of monthly payments to pre-university students, with the aim of reducing the problems of information inconsistency and answering the following research question: The development of a web application for the management of monthly payment collections at CENES pre-university, will it reduce the rate of non-payment by students?

In this sense, the methodological framework used consisted of 3 phases. In the first phase, the documentation of the requirements was carried out applying the IEEE-830 standard. In phase two, the software architecture was designed using architectural models such as Client-Server and Rest, later the solution was coded using Django ResFramework, Angular and flutter. Finally, black and white box tests were carried out, which allowed finding coding errors before the software went into production, on the other hand, functionality tests were carried out, carried out on the end user through the use of the application, the load and stress tests helped determine that the response times are adequate for the request between the client and the server.

Finally, the web application was executed in a controlled environment, evidencing that its operation significantly influences the management and collection process at pre-university CENES.

Keywords: Software Development, collection management, ICONIX methodology, Django ResFramework .

3. Introducción

La demanda de acceso a servicios basados en cobranzas necesita mejores capacidades para solventar este tipo de peticiones a través de herramientas tecnológicas [1][2], los servicios financieros que se prestan a través de los medios digitales, están generando una evolución notable, considerando así que del 89% al 92% del cumplimiento a tiempo de los pagos [3], pueden ser obtenidos a través de la gestión de cobros utilizando un aplicativo web.

En base a esta realidad, existen problemas de inconsistencia de información en las instituciones que realizan la gestión de cobros, el "Centro de Nivelación Estudiantil " CENES, también se ha visto afectado con el uso del método tradicional de cobros, ya que, en el transcurso de los últimos años, dadas las principales necesidades de gestión de cobranzas, la institución ha adquirido equipos y documentos necesarios para solventarlas, pero, por otra parte, el establecimiento ha ido creciendo y ganado popularidad entre el año 2015 a la actualidad ha existido un notable crecimiento del 20% de alumnado por año, trayendo como consecuencia el manejo de mayor información, dejando de ser sustentable el uso del método tradicional de cobros, desistiendo así de ser sostenible el control autónomo de las técnicas de gestión, dado que un deficiente control de cuentas por cobrar puede traer consigo como consecuencia el posible incumplimiento de pagos por concepto de sueldos, arriendos, además no contar con recursos necesarios en caso de existir un algún tipo de contingencia entre otros. El inconveniente principal surge en el departamento de secretaría dado que es el encargado del cobro de las mensualidades generadas a cada uno de los estudiantes, el uso del método tradicional de cobros, ha dado abertura a información duplicada, redundante o inclusive a perdida de dicha información, generando inconformidad por parte de los clientes, de esta manera se ha desaprovechado los recursos informáticos y humanos que disponen.

En este contexto, el presente Trabajo de Titulación (TT) busca resolver esta problemática a través del desarrolló de un aplicativo web para la gestión de cobros de mensualidades a estudiantes del preuniversitario CENES. El proyecto se desenvolvió en tres fases con ayuda de la metodología ICONIX: iniciando con la técnica de la entrevista para la recolección de requerimientos establecidos bajo el estándar IEEE-830, el que contempla la funcionalidad general del producto, se continuó con la descripción y prototipo de cada uno de los casos de uso, así como el diseño arquitectónico, una vista lógica y física del sistema; además, se realizó la codificación del aplicativo web para proseguir a realizar las respectivas pruebas.

Para finalizar, el presente informe se ha estructurado siguiendo la normativa propuesta por la Universidad Nacional de Loja. En la sección de Marco Teórico se detalla los conceptos fundamentales del proceso de cobro; sección de Metodología abarca el proceso que se realizó para desarrollar con éxito el TT, así como se detalla los recursos utilizados y los participantes que intervinieron. En la sección Resultados se puntualiza cada uno de los hallazgos encontrados por cada objetivo específico planteados para dar solución al objetivo general; en la sección de Discusión se redacta desde el punto de vista del autor, la interpretación de cada resultado obtenido. Sección de Conclusiones se detalla y destaca los resultados más importantes de acuerdo al desarrollo del TT, finalmente en la sección de Recomendaciones se describe cada sugerencia que se obtuvo a partir de la culminación del presente proyecto.

4. Marco Teórico

Esta sección se enfoca en una revisión bibliográfica con el fin de que el lector comprenda el contexto del resultado final del presente TT. En la primera sección se inicia con una descripción histórica sobre la evolución de la cobranza, en la segunda sección se desarrolla la contextualización de términos necesarios para la gestión de cobros y para finalizar en la tercera sección se detalla los conceptos necesarios para el desarrollo del proyecto.

4.1. Reseña histórica

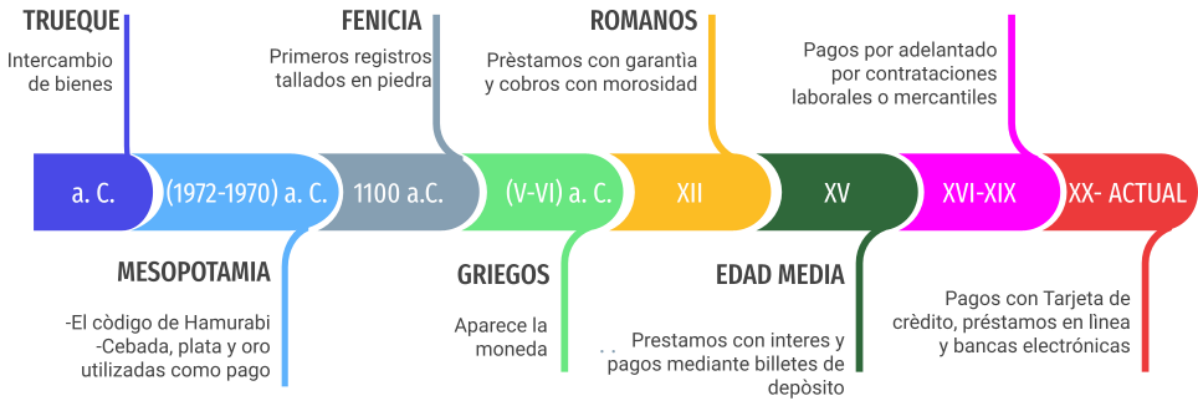


Figura 1 Historia de los cobros

En la actualidad la gestión de cobros son operaciones fundamentales dentro de los procesos de cualquier empresa, sin importar su tamaño, dado que se tiene que tratar con clientes que realizan los pagos retrasados o trimestrales, los cuales no se concilian con el flujo del efectivo real, por lo tanto, conlleva una serie de acciones, estrategias o planes con el fin de mantener el capital y que se pueda operar.

Actividades que nacieron al final de las comunidades primitivas según Arturo Morales Castro [4], son operaciones mercantiles en forma de trueque era la forma más común de comercio donde se intercambiaban bienes como frutas, comida por temporada como pago.

En la antigua Mesopotamia desde 1972-1750 a. C., se otorgaban créditos incluyendo intereses monetarios para las personas que se atrasaban en sus pagos, el pago de los créditos se realizaba mediante mercancía utilizadas como monedas como: cereales, cebada y lingotes de oro. Siendo el Código Hammurabi [5], el que reglamento el préstamo y depósitos de mercancías que se mencionaban en el contrato de la comisión.

Sin embargo, en el año 110 a. C. con los Fenicios tienen el primer antecedente en tablillas de arcilla para registrar las operaciones de crédito y cobro, siendo estos muy importantes porque

fueron una civilización que tuvieron escritura y eran excelentes navegadores por lo cual promovieron el comercio.

En el siglo V y VI a. C. en la antigua Grecia [6], Sólon consagro la supremacía del comerciante y autorizo el préstamo con intereses sin límites, además retiro a los acreedores el derecho de esclavizar a los deudores. En esta época aparece la moneda y existía un documento similar a la letra de cambio, para no portar dinero debido a las grandes distancias que recorrían a los pueblos donde realizaban operaciones de negocio internacional convirtiéndose en uno de los factores de prosperidad económica del país.

Los Romanos [7], fue una civilización muy importante debido a que da las bases a lo que son las obligaciones y del derecho civil, en este derecho se determinó que el crédito es una obligación, convirtiéndolo aún más complicado debido a que existían penalizaciones por falta de pago que iba desde confiscar los bienes, convertir al deudor en esclavo o incluso la muerte, de todas las civilizaciones Roma es la que persiste en todas sus aportaciones hasta hoy en día. Finalmente, en el siglo XII se establece los primeros bancos y es como empieza a surgir el sistema financiero.

En la edad media siglo XV el préstamo con intereses se convierte en un papel relevante de la vida económica convirtiéndose en un factor esencial del crecimiento y cambio utilizando los billetes de depósito.

En el siglo XVI- XIX aumenta el precio de la renta de los bienes raíces y el alza del precio del producto. El crédito tuvo un papel destacado en la conformación de la economía en la Nueva España, aparecen los pagos por adelantado en contrataciones laborales y mercantiles, la operación más utilizada fue cuando parte del crédito se canalizo mediante ventas al fiado difiriendo pagos. En al siglo XX hasta la actualidad se utilizan datos financieros en vez de usar vínculos familiares, en 1960 aparece la primera tarjeta de crédito, préstamos en línea y bancas electrónicas siendo instrumentos que conviven con nosotros hasta la actualidad [8].

Las empresas han ido mejorando las estrategias de cobros, en la actualidad se cuenta con soluciones tecnológicas que permiten un óptimo proceso para la gestión de cobros, con ayuda de los sistemas web de cobranza incluyendo inteligencia artificial se tiene asistencia al momento de: producir y enviar facturas automáticamente según preferencias del consumidor, examinar los hábitos de pago de los consumidores con el fin de hacer predicciones sobre las fechas de cobro, un claro ejemplo de estos aplicativos web usados para el proceso de cobranza en la actualidad son los sistemas manejados por los grandes bancos, la banca móvil es un ejemplo de servicio web que proporciona un banco con el fin de permitirle al usuario realizar acciones más rápidas como: pagos, transferencias, etc.

4.2. Cobranzas y Tablas de Amortización

Beneficios de la tecnología en la gestión de cobranzas

La gestión de cobranzas puede optimizarse mediante el uso de la tecnología, un buen desarrollo de cobranza asegura una buena salud financiera. El uso de la tecnología de la información, en el proceso de cobranzas puede administrarse mejor durante el transcurro de cobro al automatizar los pagos, generar alertas y priorizar un mayor porcentaje de recuperación de la deuda, otro beneficio es la existencia de más canales a través de los cuales gestionar la cobranza, con esto se logra la planificación de procesos individualizados para cada deudor y su tipo de deuda [9].

4.2.1. Cobranza

La cobranza es un proceso formal, mediante el cual se tramita el cobro de una cuenta pendiente por concepto de pago de un servicio, bien o producto; la labor de cobranza dentro de una institución o empresa es de gran importancia para la administración, dado que genera efectivo a partir del cobro respectivo [10].

Se considera que una venta no está totalmente realizada hasta no haber sido cobrada en su totalidad, por lo cual el cobro de una cuenta pendiente realizada a crédito, lleva la transacción a una conclusión adecuada [11].

- **Política de cobranza**

  Procedimientos que sigue una empresa o institución para cobrar cuentas vencidas, ya sea enviando constante emails, llamadas telefónicas, uso de bufetes de abogados, procedimientos judiciales, etc., a la medida que una empresa sea más restringida en el otorgamiento de crédito y más agresiva en sus procedimientos de cobro reduce sus pérdidas en cuentas incobrables; no obstante, corre el riesgo de que disminuyan sus ventas y aumenten sus gastos de cobranza al hacer un mayor uso de personal para labores de cobranza.

4.2.2. Crédito

El crédito es la posibilidad de obtener dinero, bienes o servicios sin tener que pagar al momento de recibirlos, a cambio de una promesa de pago a realizarse por el prestatario de una cantidad de dinero debidamente cuantificada en una fecha a futuro.

- **Política de crédito**

  Es un conjunto de criterios que aplican las empresas o instituciones para determinar si debe conceder el crédito a un cliente y las condiciones de otorgamiento, así como periodos promedios de pago [12]. Toda empresa que efectúe ventas a crédito debe establecer y mantener actualizada una política de crédito, con el objeto de que las decisiones individuales en este aspecto sigan un patrón consistente con las finalidades y políticas generales.

- **Descuentos en cobros**

  Es la disminución que se concede a un pago o deuda por diferentes circunstancias. Considerado como una rebaja sobre el precio original por alguna razón; los descuentos se expresan en tanto por ciento % [13].

  Los descuentos se calculan con la siguiente formula:

$$D = \frac{S * i}{100} \qquad \textit{Fórmula 1}$$

  Donde:
  D = descuento
  S = valor inicial
  i= tanto por ciento de descuento

- **Morosidad**

  Es el retraso con el cumplimiento del pago de obligaciones contraídas, por lo tanto, se considera un incumplimiento de contrato de pago en la fecha predeterminada. La morosidad es un tema de gran relevancia para las empresas o instituciones uno de los puntos principales es por el hecho de que puede provocar la insolvencia de quien padece e incluso si se generaliza, introduce un riesgo en el tejido empresarial provocando efectos perversos en cadena que perjudicarían gravemente la continuidad financiera de la empresa [14].

4.2.3. Tabla de amortización

Al contraer una deuda a crédito o con pagos periódicos, es necesario tener a conocimiento la cantidad que se debe pagar en determinado tiempo, en el campo financiero el documento que permite analizar de manera detallada estos pagos se lo conoce como tabla de amortización. Una tabla de amortización es una herramienta que hace posible establecer claramente la división de cada pago en interés y abono a la deuda, así como la cantidad que aún queda

adeudando después de cada pago [15]. Para la construcción de las tablas de amortización se realiza de la siguiente forma:

Tabla 1 Aspectos para la construcción de una tabla de amortización

Número de Pago	**Pago periódico**	**Abono a capital**	**Saldo**
Número de pago correspondiente	Indica el valor de cada uno de los pagos que son constantes	Es la parte del pago que se descuenta de la deuda	Es el valor de la deuda después de descontar el abono

Adicional para calcular el número de pagos a realizar es necesario determinar el valor de dichos pagos, esto se logra con el uso de la fórmula para calcular la renta de una anualidad vencida, la misma que se muestra a continuación:

$$R = \frac{C}{n}$$ *Fórmula 2*

Donde:

R: es la renta

C: valor actual de la anualidad

n: número de pagos

4.3. Metodologías para el desarrollo del proyecto

3.1.1. UML (Lenguaje de Modelado Unificado)

UML es un lenguaje de modelado para especificar, construir, documentar y visualizar partes de un sistema de software desde distintos puntos de vista, puede ser usado en cualquier proceso de desarrollo lo largo de todo el ciclo de vida y puede aplicarse a todos los dominios de aplicación, UML no se considera una notación propietaria, ni un proceso, ni una metodología , ni es un método, se considera una unificación de los métodos de modelado de objetos por medio de la identificación y definición de la semántica de los conceptos fundamentales y elección de una representación gráfica con una sintaxis simple, expresiva e intuitiva [16].

- **Diagramas y Vistas**

  UML define varios modelos para la representación de los sistemas que pueden verse y mediante un conjunto de diagramas diferentes, como diagramas de estructura y de comportamiento.

  Una vista es un subconjunto de las construcciones de modelado de UML que representa un aspecto del sistema, se pueden agrupar en áreas conceptuales como:

  - Estructural: Vista estática, diseño y de casos de uso
  - Dinámica: Vista de máquina de estados, actividad, y de interacción
  - Física: Vista de despliegue
  - Gestión: Vista de gestión de modelo y perfiles

- **Vista de Caso de Uso**

  La vista de caso de uso captura la funcionalidad de un sistema, subsistema o clase, tal como se muestra a un usuario exterior. Reparte la funcionalidad del sistema en transacciones significativas para los usuarios ideales de un sistema de un sistema, los usuarios del sistema se denominan actores y las particiones funcionales se conocen con el nombre de casos de uso, la técnica que se utiliza para modelar esta vista es el diagrama de casos de uso.

  Los CU son una técnica para especificar los requisitos funcionales propuesto inicialmente por Ivar Jacobson [17], e agregada a UML, modela la funcionalidad del sistema tal como la observan los funcionarios externos, denominados actores, que interactúan con el sistema desde un punto de vista individual.

  Componentes principales son:

  1. Sujeto: sistema que se modela
  2. Casos de uso: entidades externas que interactúan con el sistema
  3. Actores: entidades externas que interactúan con el sistema

3.1.2. Estándar IEEE 830

El análisis de requisitos es considerado una de las tareas más importantes dentro del ciclo de vida del desarrollo del software, dado que es ahí cuando se determina las bases de la aplicación a desarrollarse. El análisis de requisitos es un proceso de estudio de las necesidades de los usuarios para llegar a una definición de los requisitos del sistema, así como el proceso

de estudio y refinamiento de dichos requisitos, definición proporcionada por el IEEE [18]. Un requisito es una condición o capacidad que necesita el usuario para resolver un problema o conseguir un objetivo determinado.

A continuación, el esquema que sirve como ejemplo para escribir un Especificación de Requisitos de Software (ERS), no se tiene que seguir el esquema, ni utilizar los nombres dados, pero en si un ERS debe incluir la información presentada [19] en la Figura 2.

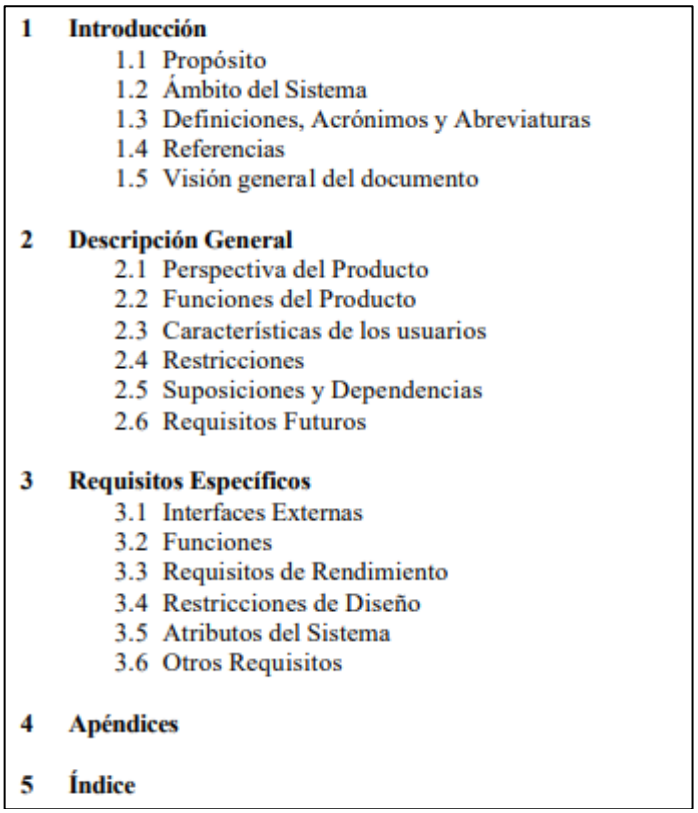


Figura 2 Estructura de una ERS

3.1.3. Metodología ICONIX

Se basa en proceso de desarrollo de software práctico simplificado en comparación con otros procesos más tradicionales, presenta de manera clara las actividades de cada etapa y exhibe a una secuencia de pasos que deben ser seguidos, unificando un conjunto de métodos de orientación a objetos con el fin de abarcar todo el ciclo de vida de un proyecto [20] [21].

ICONIX está adaptado a soporte UML (Lenguaje de Modelado Unificado**),** sus características fundamentales son:

Figura 3 Características de la metodología ICONIX

Las tareas que se deben realizar dentro de la metodología ICONIX son:

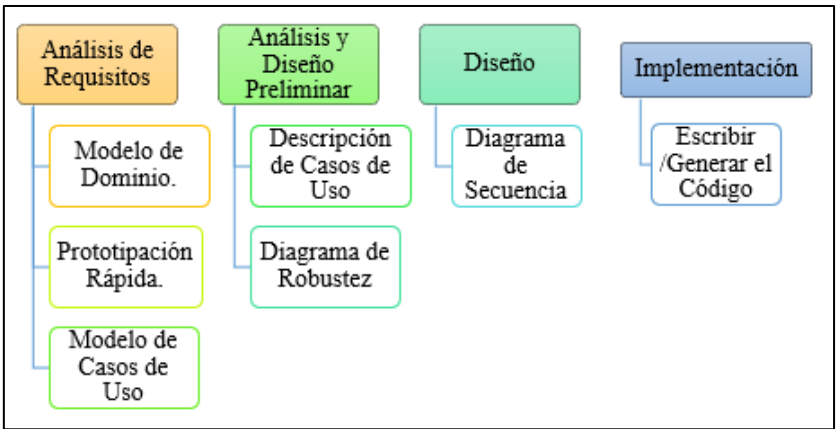


Figura 4 Tareas de la metodología ICONIX

- **Fase 1: Análisis de Requerimientos**

  Se realiza un relevamiento de todos los requisitos que en principio deberían ser parte del producto a desarrollar.

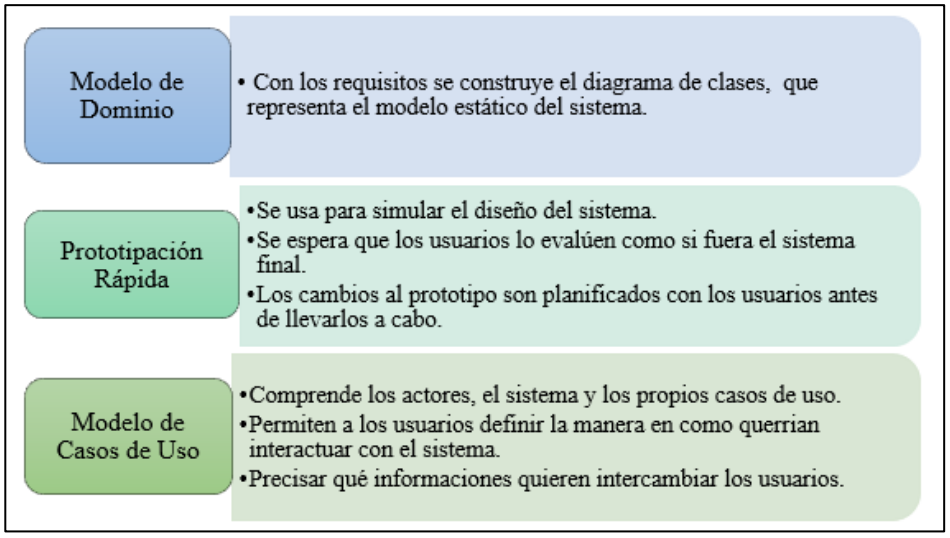


Figura 5 Análisis de Requisitos ICONIX

- **Fase 2: Análisis y Diseño Preliminar**

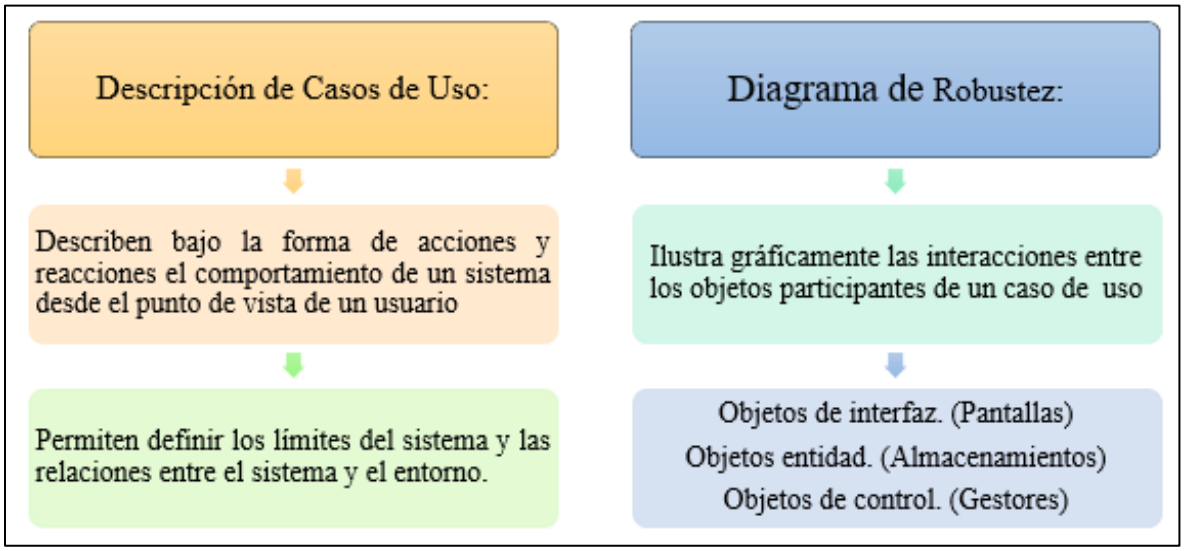


Figura 6 Análisis y Diseño Preliminar ICONIX

- **Fase 3: Diseño**

  Diagrama de secuencia

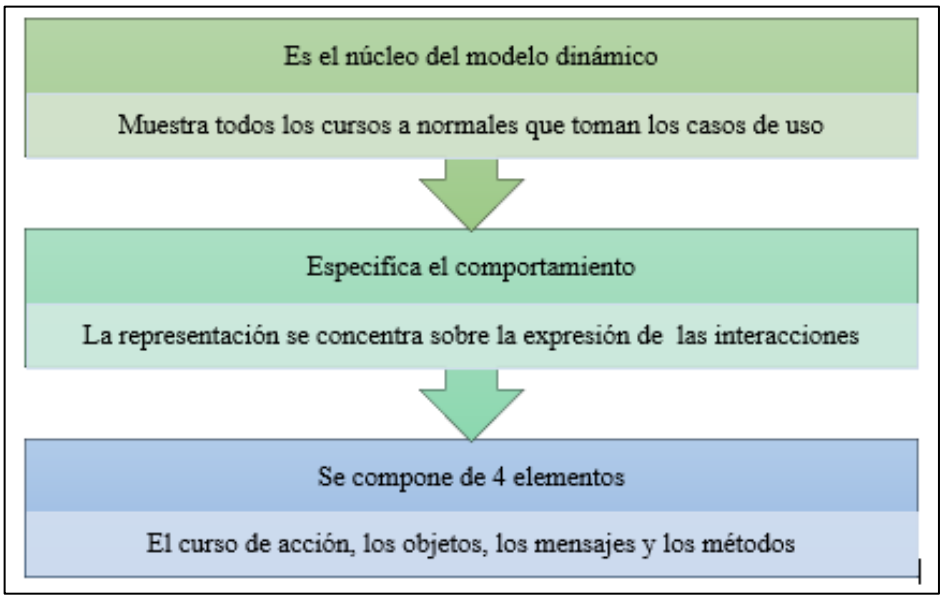


Figura 7 Diseño ICONIX

- **Fase 4: Implementación**

  Escribir/Generar código

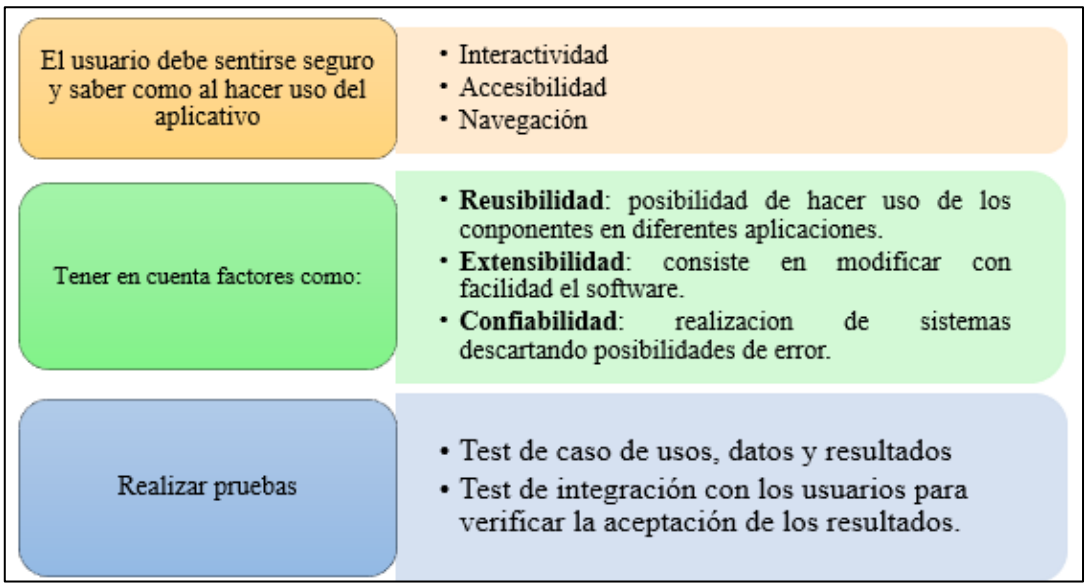


Figura 8 Implementación ICONIX

4.4. Tecnologías para el desarrollo del proyecto

4.4.1. Python

Python es un lenguaje de programación que existe desde los años 1990s [22], es sencillo, legible y elegante, una de las razones de su éxito es el contar con una licencia de código abierto permitiendo su uso en cualquier escenario [23]. Como cualquier otro lenguaje de programación que tiene sus características propias, Python cuenta con las suyas:

- Es un lenguaje interpretado, no compilado.
- Tipado dinámico, una variable puede tomar valores de distinto tipo.
- Fuertemente tipado, para un cambio de tipo se debe hacer una conversión explicita.
- Multiplataforma, dado que un código escrito en macOS funciona tranquilamente en Linux o Windows y viceversa.

Se usa Python para diversos fines como:

- **Desarrollo Web**: Contiene diferentes Frameworks que permiten desarrollar páginas web a todos los niveles, como Django, Pyramind, Flask o Bottle.
- **Ciencia y Educación:** Tiene librerías como SciPy o Pandas, lo que convierten en una herramienta perfecta para enseñar conceptos de programación.

- **Desarrollo de Interfaces Gráficos:** Kivy o pyqt son librerías que se usan para el desarrollo de interfaz gráfica de usuario.
- **Desarrollo Software:** Es usado por parte de los desarrolladores, para testing.
- **Machine Learning:** contiene librerías de aprendizaje automático como Keras, TensorFlow, PyTorch o sklearn.
- **Visualización de Datos:** matplotlib, seaborn o plotly, son las librerías más usadas para mostrar datos en gráficas.
- **Finanzas y Trading:** QuantLib o qtpylib son librerías de fácil uso, por lo cual cada vez es más usado en estos sectores.

- **Framework Django**

  Es un marco web Python de alto nivel que incentiva al desarrollo rápido, diseño limpio y programático. Django ha sido diseñado con el fin de ayudar a los desarrollares a llevar aplicaciones desde el concepto hasta la finalización lo más rápido posible, el desarrollador se concentra en escribir un código limpio y de fácil mantenimiento [24][25]. Se encarga de la autenticación del usuario, administración de contenido, en otras tareas, desde el primer momento, así mismo se encarga de ayudar a los desarrollad ores a evitar algunos errores de seguridad, su sistema de autenticación proporciona una forma segura de administrar las cuentas y contraseñas de los usuarios.

- **Django REST Framework**

  Es un conjunto de herramientas de Django que permite la construcción de proyectos de software bajo la arquitectura REST, para crear API's web [26].

  El marco REST permite:

  - Una API navegable por la Web es una gran ganancia de usabilidad para sus desarrolladores.
  - Serialización que admite fuentes de datos ORM y no ORM .
  - Personalizable hasta el final.
  - Amplia documentación y gran apoyo de la comunidad .
  - Utilizado y confiado por empresas reconocidas internacionalmente, como Mozilla , Red Hat , Heroku y Eventbrite .

4.4.2. Base de Datos PostgreSQL

En la actualidad el uso de base de datos es de gran utilidad dado que sirven para gestionar grandes ficheros y facilitar la consulta de información definiendo un esquema de permisos para que personas o programas puedan acceder a esos datos presentando la información requerida de forma detallada, adecuada y clara [27].

PostgreSQL Es un sistema de gestión de base de datos objeto-relacional distribuido bajo licencia BSD y con su código de fuente disponible libremente siendo el más potente del mercado, utiliza un modelo cliente -servidor y utiliza multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en los procesos no afecta al resto y el sistema continuara funcionando [28] [29].

Según el sitio oficial de PostgreSQL ofrece características modernas como:

- 100% ACID (conjunto de propiedades que garantizan que las transacciones de una Base de Datos se procesan de manera fiable)
- Consultas complejas (subconsultas, joins, etc.)
- Vistas actualizables
- Integridad referencial (claves primarias y foráneas)
- Restricciones (Constraints)
- Disparadores (triggers)
- Vistas (views)
- Reglas (rules)

Además, el usuario puede ampliar de muchas maneras, agregando nuevos

- Tipos de datos
- Funciones
- Operadores

4.4.3. Angular

Angular es una plataforma que permite desarrollar aplicaciones web utilizando JavaScript y HTML en la parte del cliente, permite la creación de aplicaciones web de una sola página (SPA: single-page-application) donde realiza la carga de los datos de manera síncrona. Gracias al uso de componentes, se puede encapsular mejor la funcionalidad facilitando el mantenimiento de las aplicaciones [30].

4.4.4. Flutter

Flutter es un kit de desarrollo software de interfaz gráfica open source creado por Google, que ofrece un gran número de bibliotecas. Es utilizado actualmente en el desarrollo de

aplicaciones para Android, iOS, Windows, Mac, Linux, Google Fuchsia y web a partir de un único código base [31]. El proceso utilizado es la compilación del código en lenguaje Dart a código nativo de cada plataforma.

Flutter se usa principalmente para desarrollar aplicaciones de Android y iOS sin necesidad de escribir un código base propio para cada uno de estos sistemas, completamente diferentes entre sí. En este contexto, las aplicaciones móviles se ejecutan como auténticas aplicaciones nativas en los dispositivos. Antes de su publicación, se compilan para la plataforma correspondiente, de manera que no necesitan un módulo runtime ni un navegador. Sobre la misma base de código, se pueden crear aplicaciones web para navegadores y programas nativos para Windows, Linux y macOS [32].

Google usa Flutter, por ejemplo, para distintos módulos de Google Assistant y la interfaz de usuario de Google Home Hub.

4.4.5. Postman

Postman nace como una herramienta que principalmente nos permite crear peticiones sobre APIs de una forma muy sencilla y, de esta manera, probar las APIs. Todo basado en una extensión de Google Chrome. El usuario de Postman puede ser un desarrollador que esté comprobando el funcionamiento de una API para desarrollar sobre ella o un operador el cual esté realizando tareas de monitorización sobre un API [33].

Características de Postman

- **Crear Peticiones,** te permite crear y enviar peticiones http a servicios REST mediante una interface gráfica. Estas peticiones pueden ser guardadas y reproducidas a posteriori.
- **Definir Colecciones**, mediante Postman se puede agrupar las APIs en colecciones. En estas colecciones podemos definir el modelo de autentificación de las APIs para que se añada en cada petición. De igual manera se puede ejecutar un conjunto de test, así como definir variables para la colección.
- **Gestionar la Documentación**, genera documentación basada en las API y colecciones que se ha creado en la herramienta. Además, esta documentación se puede hacer pública.
- **Entorno Colaborativo**, permite compartir las API para un equipo entre varias personas. Para ello se apoya en una herramienta de colaborativa en Cloud.

- **Genera código de invocación**, dado un API es capaz de generar el código de invocación para diferentes lenguajes de programación: C, cURL, C#, Go, Java, JavaScript, NodeJS, Objective-C, PHP, Python, Ruby, Shell, Swift
- **Establecer variables**, con Postman se puede crear variables locales y globales que posteriormente se usen dentro de nuestras invocaciones o pruebas.
- **Soporta Ciclo Vida API management**, desde Postman se puede gestionar el ciclo de vida del API Management, desde la conceptualización del API, la definición del API, el desarrollo del API y la monitorización y mantenimiento del API.
- **Crear mockups**, mediante Postman se puede crear un servidor de mockups o sandbox para que se puedan testear las API antes de que estas estén desarrolladas.

4.4.6. Jmeter

La aplicativo Apache JMeter es un software de código libre que está desarrollado 100% en lenguaje de programación Java.

JMeter fue diseñado para realizar pruebas de carga en servidores o aplicativos Web por medio del protocolo HTTP, pero debido a su gran popularidad, se expandió para incluir otros protocolos de comunicación. JMeter fue desarrollado por Stefano Mazzocchi para cubrir la necesidad de realizar pruebas de carga y estrés al proyecto Apache JServ, el cual fue reemplazado tiempo después por el proyecto Apache Tomcat. En noviembre del 2011, JMeter fue reconocido como un proyecto de alto nivel para la fundación Apache y es por ello que se le asignó un sitio web exclusivo [34].

Hoy en día es considerada la herramienta de carga más popular y es por ese motivo que nosotros queremos trabajar en su promoción, correcta utilización y mentoría. JMeter es la herramienta más popular para realizar pruebas de carga. En un principio fue diseñada para realizar las pruebas de stress centradas en las aplicaciones web, pero actualmente se permite diseñar pruebas para bases de datos, FTP, prácticamente para cualquier cosa.

Otra de sus principales características es su diversidad de pruebas. JMeter permite realizar desde un componente muy sencillo (solicitud) hasta secuencias complejas que permiten analizar el comportamiento de la aplicación. Puede manipular y almacenar la secuencia para su reutilización en otras pruebas.

4.5. Trabajos Relacionados

Es de gran importancia conocer diferentes estudios que están relacionados con la gestión de cobros, dado que permiten obtener diferentes puntos de vista, en esta sección se resumen 5 trabajos que abordan el objeto de estudio desde la perspectiva de otros autores.

"SISTEMA WEB PARA EL PROCESO DE COBRANZA EN LA EMPRESA DE CRÉDITOS SEBASTIÁN"

El presente trabajo establece como precedente la problemática de la cobranza, los mismos que se daban con procesos arcaicos, los cuales generaban estancamiento de capital y movimiento económico. Para dar solución al problema antes mencionado, utiliza UWE para el planteamiento de actividades y modelado orientado a un producto web, además del uso de JAVA con su framework Java Server Faces, con base de datos MySQL y NetBeans como IDE de desarrollo. Obteniendo resultados de mejora en el cumplimiento de cobros, así como en el crecimiento de la empresa. Con la investigación realizada en el trabajo de titulación ha quedado demostrado que la aplicación de nuevas tecnologías de sistemas de información ayuda a mejorar procesos informáticos, haciendo que sean más competitivos y manteniendo el orden al manejo de información empresarial [35].

"ANÁLISIS, DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL PARA REGISTROS Y COBRO DE MATRÍCULA Y PENSIONES PARA LA UNIDAD EDUCATIVA PARTICULAR MIXTA MERCEDES DE JESÚS MOLINA MEDIANTE UN APLICATIVO WEB."

El proyecto mencionado anteriormente tiene como objetivo, realizar el análisis, diseño, desarrollo e implementación de un Sistema de control para registro y cobro de matrícula y pensiones, en la escuela Mercedes de Jesús Molina. La principal idea para la creación del proyecto ha sido el mejorar los procesos que se desarrollan en la escuela en cuanto al registro de cobro de matrículas, con el fin de promover en mayor índice de rendimiento minimizando los tiempos de ejecución en relación a los procesos de atención al cliente. El proyecto se desarrolló con el uso de UML, PHP como lenguaje de programación, con motor de base de datos MySQL, con diseño orientado a MVC, de esta manera se determina que el uso de

aplicativos webs permite a las instituciones mejorar la atención al representante y mejorar su imagen como institución [36].

“SISTEMA WEB PARA EL CONTROL DE PAGOS EN LA I.E.P. DIEGO THOMSON DE MANGOMARCA, S.J.L 2017”

El trabajo de titulación menciona que para la creación del sistema web se consideró como antecedente a las deficiencias en la cantidad de personas atrasadas en el pago, recibos perdidos, tiempo del registro de pagos, así como de la elaboración de reportes de los mismos. El principal objetivo, contribuir a la reducción de tiempo en los registros de pago como elaboración de reportes, reducir el gasto en papelería, cobrar mensualidades a tiempo y reducción de recibos perdidos. El sistema web se desarrolló con el uso de la metodología Iconix, la cual propone un desarrollo ágil como XP, realizando la respectiva documentación haciendo uso de la notación UML, además el lenguaje de programación PHP y el motor de base de datos MySQL. Logrando como resultado reducir el tiempo de pagos en un 51,30%, tiempo de elaboración de reportes de pagos en 93,90%, costo de uso de papelería 66,67%, logrando así efectos positivos en la institución [37].

"DESARROLLO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB PARA LA ADMINISTRACIÓN DE PAGOS DE AGUA DEL BARRIO EL ZARZA-CANTÓN YANTZAZA"

El objetivo principal para el desarrollo del trabajo de titulación consiste en desarrollar e implementar una aplicación web para la Junta Directiva de Agua del Barrio El Zarza – Cantón Yantzaza, con el fin de resolver las necesidades de los usuarios, así como optimizando tiempo y trabajo a la Junta Directiva. La aplicación Web, se desarrolló mediante el uso de la metodología ICONIX, para el modelado Enterprise Architect, Bootstrap para FrontEnd, Python como lenguaje de programación con el uso del framework Django, Postgres para la base de datos, del lado del servidor Nginx y Hostinger para el hosting. Mediante las pruebas de funcionalidad realizadas al aplicativo web se pudo verificar la eficiencia, arrojando resultados esperados para la satisfacción del usuario final [38].

5. Metodología

5.1.Área de Estudio

El presente Trabajo de Titulación (TT), se desarrolló durante el periodo Octubre-Febrero2023, en la Institución del preuniversitario CENES, con la colaboración de los interesados: Ing. Cristhian Salazar y Sra. Guadalupe Hurtado, quienes brindaron la información necesaria para el desarrollo del proyecto.

5.2.Procedimiento

Con el fin de dar el cumplimiento debido del objetivo general "Implementar un aplicativo web para la gestión de cobros de mensualidades en el preuniversitario CENES.", del TT se plantearon actividades por cada objetivo específico, los cuales se detallan a continuación:

1. Documentar los requerimientos solicitados por la institución para el desarrollo del aplicativo web.

Para el desarrollo del primer objetivo se utilizó las técnicas de elicitación de la entrevista, prototipos y escenarios con el objetivo de conocer a fondo el dominio o procesos de cobros que se realiza en la empresa por parte de los Stakeholder involucrados. Además, con el uso de la metodología ICONIX estableció un conjunto de actividades para cumplir con el ciclo de vida del software (análisis de requerimientos, análisis y diseño preliminar, diseños y la implementación).

1.1. Identificar las diferentes etapas del proceso de cobro de mensualidades mediante una entrevista a la secretaría/contadora que es la principal responsable de los cobros.

Se aplicó la técnica de la entrevista en una ocasión a la Sra. Guadalupe Hurtado que es la persona encargada de los cobros en la Institución, mediante la cual se pudo determinar el proceso actual que manejan para realizar cobros así mismo se identificó las necesidades. El detalle amplio de la aplicación de la técnica véase Anexo 1.

1.2. Especificar los requisitos.

Para la documentación de la primera fase de la metodología de la metodología ICONIX, se utilizó el estándar IEEE-830 Véase Anexo 2, donde se determinó los actores, requerimientos funcionales y no funcionales que deberá poseer el aplicativo web. De la

misma manera, para que el interesado tenga una mejor perspectiva del producto final se realizó los diagramas de casos de uso.

1.3. Validar los requisitos con la persona responsable del proceso de cobros.

Para la validación de cada uno de los requisitos se realizó una reunión con los interesados, con el fin de exponer y dar a conocer cómo funcionará el aplicativo web. Véase Anexo 1.

2. Diseñar el modelo arquitectónico en base a los requerimientos con el fin de obtener la interfaz gráfica del aplicativo web.

En el segundo objetivo se desarrolló el modelo arquitectónico que tendrá el aplicativo web para la gestión de cobros del preuniversitario CENES, se emplea la segunda y tercera fase de la metodología ICONIX.

Para al diseño arquitectónico se realizó el análisis y diseño preliminar, donde se realizó la especificación de los casos de uso y un prototipado de cada uno de ellos.

2.1. Diseñar el estilo arquitectónico

Para diseño del estilo arquitectónico se tuvo en cuenta características específicas como: el cliente final, la tecnología a desarrollar y el gestor de base de datos. Obteniendo de esta manera una imagen que representa la funcionalidad del aplicativo web.

2.2. Generar los artefactos de diseño

Para una mejor comprensión de los datos y de las relaciones que tendrán esto se desarrolló el diagrama de dominio, donde se especifica cada una de las tablas y su relación.

2.3. Realizar los diagramas de secuencia y clases

Tomando como base los diagramas de robustez obtenidos en la fase de Análisis y diseño Preliminar, se procedió con el desarrollo de los diagramas de secuencia de cada uno de los requisitos del aplicativo web, véase Anexo 5.

2.4. Realizar los diagramas de componente y despliegue

Una vez identificado el estilo arquitectónico, el modelo del dominio y los diagramas de secuencia, se procedió con el diseño de los diagramas de componentes y despliegue que abarca nuestro aplicativo web.

3. Evaluar el aplicativo web en un ambiente simulado.

En el tercer objetivo se emplea la fase de implementación de la metodología ICONIX.

Previo al diseño arquitectónico se realizó la codificación de la solución informática, usando el lenguaje de programación Python, con su framework Django Rest, terminada la codificación se continuó con la etapa de las pruebas, para verificar que el sistema final cumpla con lo requerido y opere de manera exitosa.

3.1. Pruebas Caja Negra

Se aplicó la prueba de caja negra al código de la solución informática con el fin de verificar sus funcionalidades, para la aprobación de cada resultado se contempló que se realice con éxito cada uno de los requisitos funcionales planteados en el objetivo 1, para un mejor detalle revisar anexo 6.

3.2. Pruebas de Caja Blanca

Para comprobar el comportamiento interno y la estructura del programa, se emplea la prueba de caja blanca, con el fin de rastrear posibles rutas de ejecución a través del código fuente y así determinar que valores de entrada forzarían la ejecución de esas rutas, identificando algunos tipos de debilidades en el diseño del código. Para mejor detalle revisar el anexo 7.

3.3. Pruebas Funcionales

Para evaluar y verificar que el producto final cumple con lo que debe hacer, se realizó la prueba funcional de aceptación hacia el usuario, el que se encargada de revisar que la solución funciona según lo previsto. Para un mejor detalle de estas pruebas revisar el anexo 8.

5.3.Recursos

A continuación, se detalla los diferentes recursos, que se utilizó para el desarrollo de TT.

5.3.1. Métodos

- **Bibliográfico**

  Se uso el método bibliográfico para recolectar la información que ayude con los conceptos de marco contextual y conceptual, identificando literatura gris. En una primera parte se definió, obtener información de base de datos confiables, se usó

IEEE, Google Scholar y además repositorios bibliográficos de universidades (tesis, artículos), de esta manera se adquirió información estructurada, confiable y verídica.

- **Metodología de desarrollo de software**

  Se utilizó la metodología ICONIX, para el desarrollo de todo el proceso para la creación del aplicativo, desde la especificación de requerimientos, hasta las pruebas del aplicativo (revisar apartado 6.1).

5.3.2. Técnicas

- **Entrevista**

  Para identificar el proceso para la gestión de cobros, se aplicó la técnica de la entrevista semiestructurada (ver anexo 1), la misma que se realizó mediante un guión preestablecido, secuenciado y dirigido a la persona encargada del cobro de mensualidades en el preuniversitario CENES.

  Adicional a la identificación del proceso, la entrevista sirvió para identificar y nombrar a los actores del proceso, mediante el uso de diagramas UML como son los casos de uso.

- **Observación**

  Se aplicó la técnica de la observación no estructurada, con el fin de observar los resultados obtenidos por cada objetivo y así valorar el estado de estos, si su funcionamiento es correcto o tiene que ser reestructurados o modificados.

- **Reuniones**

  Con el fin de presentar avances y aprobaciones de las diferentes etapas por las que pasa el desarrollo del aplicativo web se empleó la técnica de reuniones.

5.3.3. Estándares

- **IEEE 830**

  Para determinar los requisitos se utilizó el estándar IEEE 830, en este documento se detalla los autores involucrados, los requerimientos funcionales y no funcionales, redactados en base a la información brindada en la entrevista y reuniones.

5.3.4. Recursos Técnicos

En la tabla 2 presentada a continuación, se detalla las herramientas que fueron necesaria para el desarrollo del TT:

Tabla 2 Componentes Software

Componente	Descripción
VisualStudio	• Codificación y desarrollo del aplicativo web.
Enterprise Architect	• Modelar los diagramas de Casos de Uso, secuencia, robustez.
Mendeley	• Para referenciar correctamente las de citas bibliográficas.

5.3.5. Recursos Hardware

En la tabla 3 se presentan los recursos hardware usados para el desarrollo del TT:

Tabla 3 Componentes Hardware

Componente	Descripción
Computadora	• Redacción del TT. • Modelamiento Arquitectónico.

5.4.Participantes

El presente Trabajo de Titulación fue ejecutado por los siguientes participantes:

- Carla Isabel Troya Capa, estudiante a cargo de la ejecución del TT.
- Ing. Edwin René Guamán Quinche, docente que dirigió y asesoró el desarrollo del TT.
- Ing. Cristhian Salazar, director del preuniversitario CENES, cliente que solicitó el desarrollo de la solución informática.
- Sra. Guadalupe Hurtado, secretaria del preuniversitario CENES, participante en la entrevista y en las pruebas de usabilidad.

6. Resultados

Se describen cada uno de los resultados obtenidos por cada objetivo específico con el uso de la metodología ICONIX, esta sección se divide en tres fases: en la sección 6.1. se exponen los resultados obtenidos el primer objetivo, detallando el proceso realizado para la obtención de los requerimientos del software. En la sección 6.2. se evidencia el diseño arquitectónico con el que cuenta el aplicativo web y se detalla la codificación empleada para la realización de la solución informática. Finalmente, en la sección 6.3. se especifica cada tipo de prueba realizada al aplicativo web.

6.1. OBJETIVO 1: Documentar los requerimientos solicitados por la institución para el desarrollo del aplicativo web.

Para dar cumplimento al primero objetivo se llevó a cabo la aplicación de la técnica de la entrevista a la Sra. Guadalupe Hurtado, encargada de realizar los cobros en la Institución. Además, mediante el uso de la metodología ICONIX, en la fase de análisis de requisitos, se realizó la determinación de los requerimientos con el uso del estándar IEEE 830, en el cual se detalló los principales usuarios, la perspectiva del producto y los requerimientos funcionales y no funcionales.

6.1.1. Identificar las diferentes etapas del proceso de cobro de mensualidades mediante una entrevista a la secretaría/contadora que es la principal responsable de los cobros.

Para obtener la información del proceso de gestión de cobros dentro del preuniversitario “CENES”, se aplicó la técnica de la entrevista semiestructurada a la Sra. Guadalupe Hurtado, encargada de realizar los cobros en la Institución. En base a la aplicación de esta técnica de elicitación se documentó el estado actual de los procesos y las necesidades existentes. El anexo 1, se detalla la aplicación de la entrevista.

En el mismo contexto, en la Figura 9, se ilustra las actividades que se realizan el proceso de gestión de cobros. La persona encargada de realizar los cobros debe primero verificar si el estudiante está matriculado, en el caso de que lo este, debe revisar la ficha para saber que cuota debe cancelar o si está atrasado, en caso de que no se encuentre matriculado debe cumplir con todo el proceso manual de matriculación.

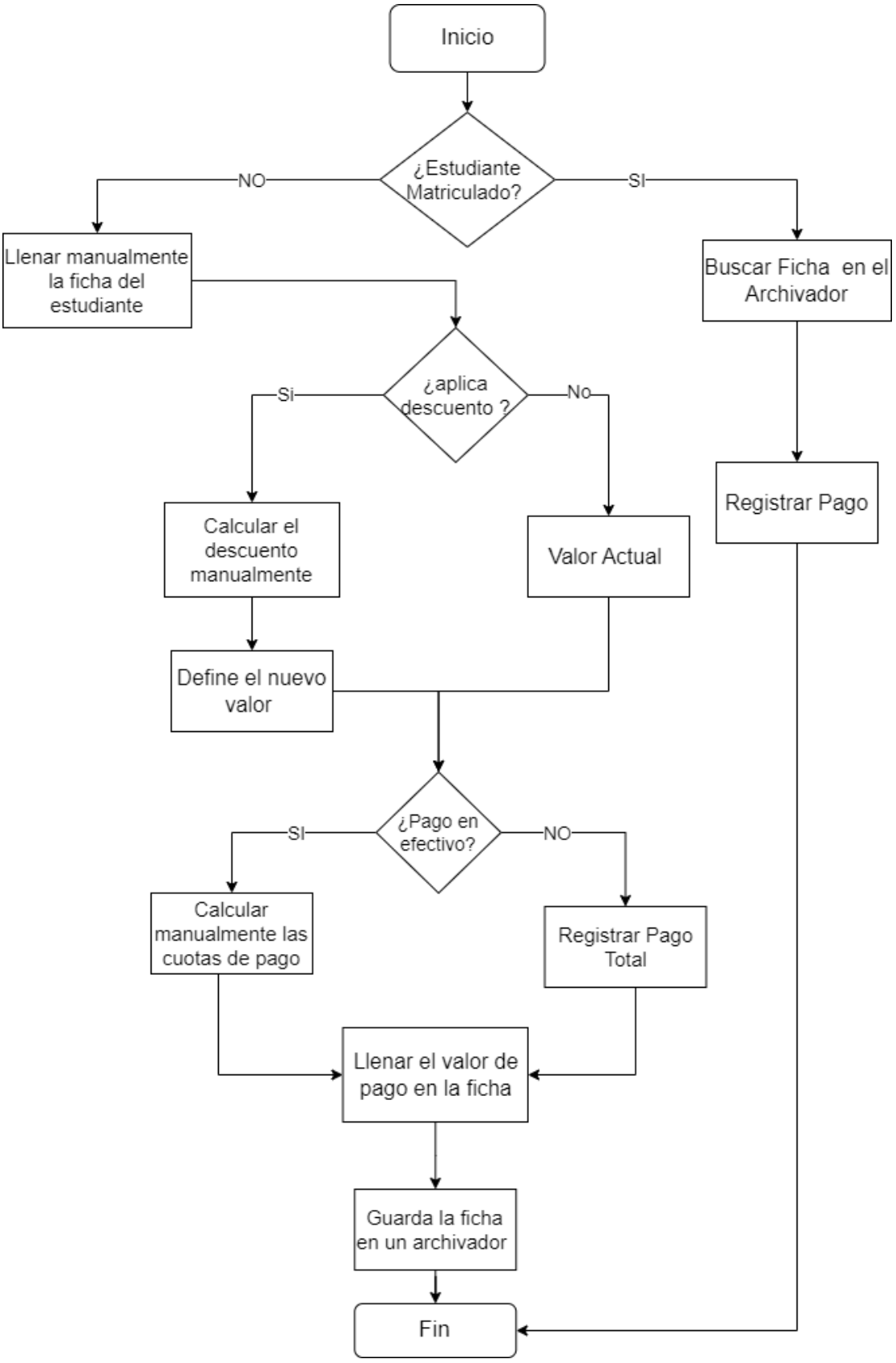


Figura 9 Diagrama de flujos de gestión de cobros

6.1.2. Especificación de los requisitos

Para la documentación de requisitos de software del proyecto se aplicó el estándar IEEE-830, donde se detalla los principales actores, requisitos funcionales y no funcionales. En el anexo 2 se detalla el documento final de especificación de requisitos.

Fase 1: Análisis de Requisitos

- **Diccionario de Términos**

Tabla 4 Diccionario de Términos

Nombre	Descripción
Aplicativo web	Es un conjunto de páginas dinámicas cuyo contenido se determina después que un usuario haya interactuado con ella accediendo a un servidor web a través de Internet o de una intranet mediante un navegador.
Usuario	Persona que usará el sistema para gestionar el proceso
ERS	Especificación de Requisitos Software
RF	Requerimiento Funcional
RNF	Requerimiento No Funcional
CENES	Centro de Nivelación Estudiantil
COHORTE	Son las opciones que tendrá un curso (Mañana, Tarde, etc.)

- **Usuarios o Actores**

  Se identificó tres usuarios que interactúan en el proceso de cobros: administrador, secretaria y estudiante

Tabla 5 Tipo de usuario: "Administrador"

Tipo de usuario	Administrador
Formación	Educador
Actividades	Control y manejo del sistema según su usuario asignado

Tabla 6 Tipo de Usuario: "secretaria"

Tipo de usuario	Secretaria
Formación	Contadora
Actividades	Control y manejo del sistema según su usuario asignado

Tabla 7 Tipo de Usuario: "Estudiante"

Tipo de usuario	Estudiante
Formación	Estudiante
Actividades	Manejo del sistema según su usuario asignado

- **Requerimientos Funcionales**

  En la tabla 8 se detallan 9 requisitos funcionales de software. Los requisitos con prioridad alta y media serán implementados en la primera versión del software.

Tabla 8 Requerimientos Funcionales

REQUERIMIENTOS FUNCIONALES		
#	**Descripción**	**Prioridad**
RF001	Iniciar sesión	Alta
RF002	Gestionar curso	Alta
RF003	Gestionar cohorte	Alta
RF004	Crear un estudiante	Alta
RF005	Editar datos del estudiante	Alta
RF006	Matricular estudiante	Alta
RF007	Registrar pago	Alta
RF008	Editar pagos	Media
RF009	Visualizar Reporte	Media
RF010	Visualizar las matrículas	Media

- **Requerimientos No Funcionales**

  Por otra parte, se han identificado 4 requerimientos no funcionales detallados en la tabla 9.

Tabla 9 Requerimientos No Funcionales

REQUERIMIENTOS NO FUNCIONALES		
#	**Categoría**	**Descripción**
RNF001	USABILIDAD	El aplicativo web debe ser “responsive” para garantizar la adecuada visualización en diferentes dispositivos como computadoras, Tablet y teléfonos inteligentes.
RNF002		La interfaz del aplicativo debe ser amigable para el usuario con el fin que le permita el fácil uso.
RNF003		El tiempo de aprendizaje del software deberá ser menor a 3 horas.
RNF004	DESEMPEÑO	El aplicativo debe garantizar el desempeño en cuanto a información consultada y/o actualizada que sea de forma permanente y simultáneamente.

- **Diagrama de Casos de Uso**

  En la Figura 10 se presenta el CU General, en el que se muestra las principales funcionalidades y la interacción con los actores correspondientes del sistema web propuesto.

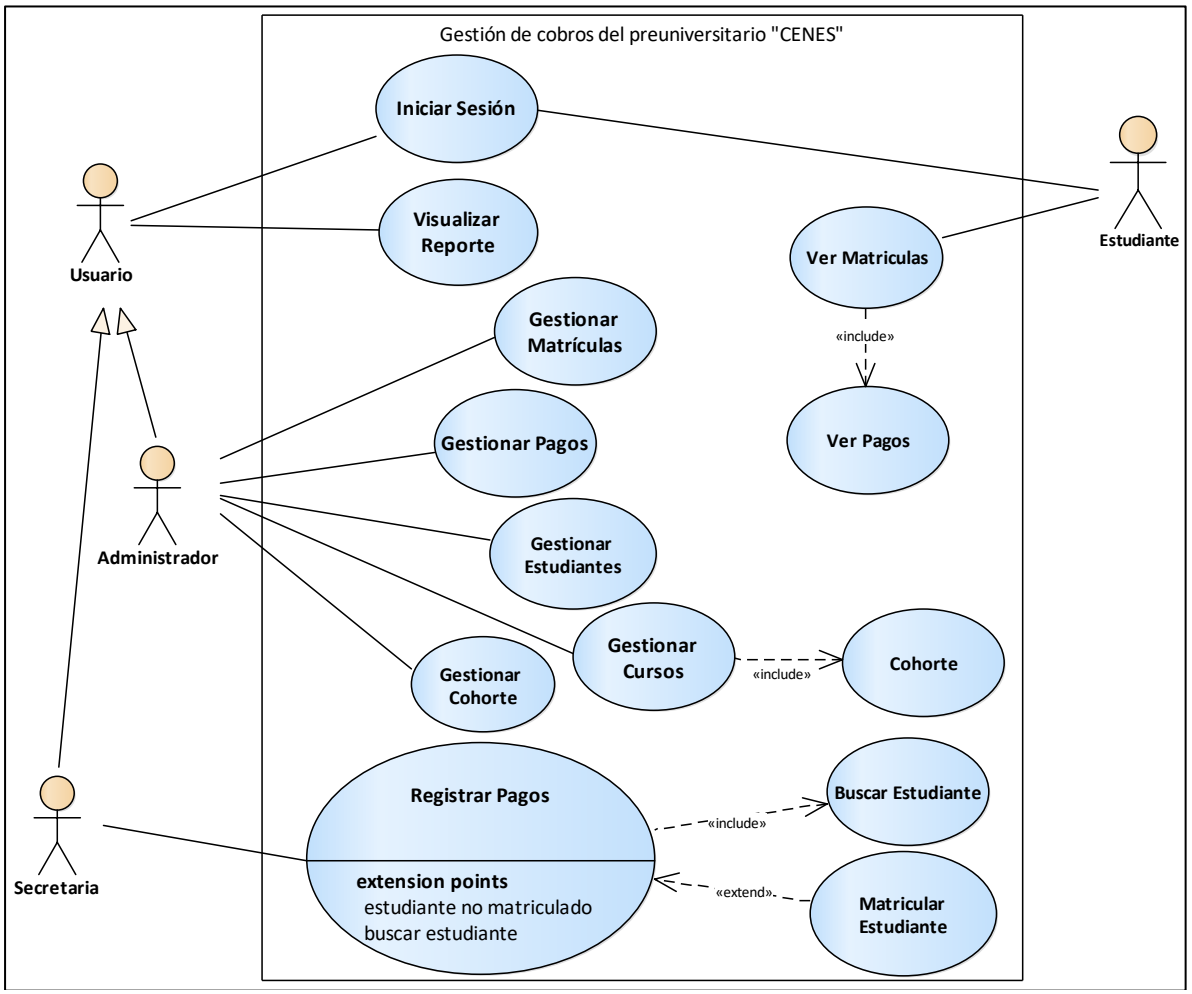


Figura 10 Caso de uso General para el proceso de cobros

- El usuario es el actor que tiene ingreso al inicio de sesión, matrícula y visualización del reporte, por lo que el actor administrador y el actor secretaria heredan esas funcionalidades.
- El actor administrador puede gestionar todo el sistema web, como son los cursos, estudiantes, matrículas y pagos, mientras que el actor secretaria solo tiene acceso a ciertas actividades, como el registro de un pago, registro de una matrícula o el registro de un estudiante.

- **Modelo del dominio inicial**

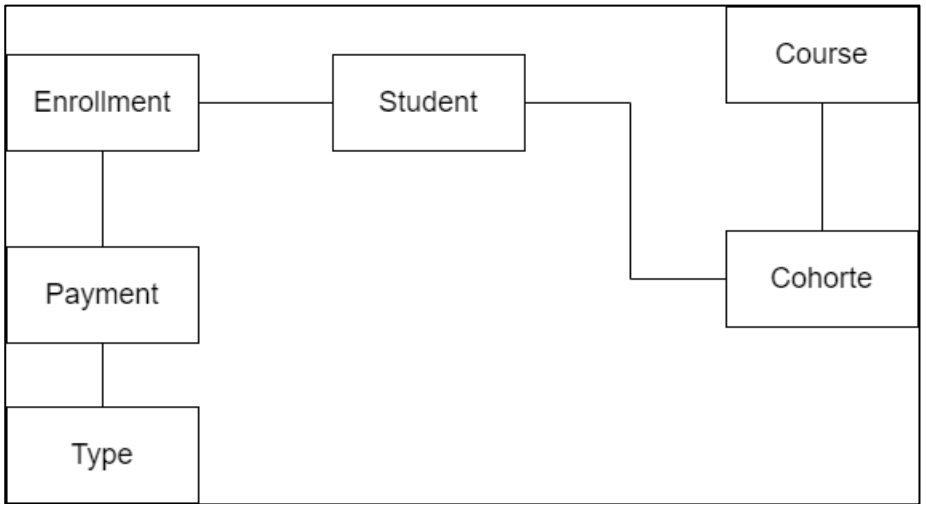


Figura 11 Modelo del Dominio Inicial

6.1.3. Validar los requisitos con la persona responsable del proceso de cobros

Para la verificación y validación de los requisitos del sistema web se realizó una reunión presencial con la persona encargada el proceso de cobros en la institución, la Sra. Guadalupe Hurtado, donde se presentó la especificación de los requerimientos propuestos, los casos de uso. En la primera versión del proceso hubo cambios sugeridos por parte de la persona encargada**,** una vez agregados los cambios se socializo el proceso en una segunda versión, la cual fue aprobada por los Stakeholder.

6.2.OBJETIVO 2: Diseñar el modelo arquitectónico en base a los requerimientos con el fin de obtener la interfaz gráfica del aplicativo web.

En esta sección se elabora el diseño arquitectónico de software para la gestión de cobros, tomando como base los requisitos y los diagramas obtenidos en el objetivo 1.

Previo al diseño arquitectónico se realizó la fase 2 de la metodología: análisis y diseño preliminar, donde se realizó la especificación de los casos de uso con un prototipado de cada uno de ellos y un diagrama de robustez con el fin de que el usuario final tenga claro lo que debe hacer y cómo hacerlo.

Fase 2: Análisis y Diseño Preliminar

Obtenidos los requisitos funcionales y el diagrama de casos de uso, aplicando los conceptos de la metodología ICONIX, se realizó la descripción de cada uno de los casos de uso, para mejor detalle de la descripción de casos de uso ver anexo 3.

En la tabla 10 se describe de manera general las tareas asociadas a todos los casos de uso general del proceso, con el fin de tener una mayor comprensión de la secuencia de dichas actividades.

Tabla 10 Descripción de los Casos de Uso

N°	Actor	Nombre	Descripción
CU01	Usuario	Iniciar Sesión	El usuario inicia sesión ingresando sus datos en el sistema.
CU02	Usuario	Visualizar Reporte	El usuario puede consultar y visualizar los resultados sobre los cobros realizados o por realizar.
CU03	Usuario	Gestionar Matriculas	El usuario puede crear una matrícula o eliminarla dependiendo del caso presentado.
CU04	Administrador	Gestionar Pagos	El usuario puede registrar, buscar un pago.
CU05	Administrador	Gestionar Estudiantes	El usuario puede crear, editar, buscar un estudiante.
CU06	Administrador	Gestionar Cursos	El usuario puedo crear, editar, buscar, eliminar un curso.
CU07	Administrador	Gestionar Cohorte	El usuario puedo crear, editar, buscar, eliminar una cohorte.
CU08	Secretaria	Registrar Pagos	El usuario puede buscar un estudiante para registrar un pago, en el caso de que no exista el estudiante, puede registrarlo.
CU09	Estudiante	Ver Matrículas	El usuario podrá visualizar los cursos en los que se encuentra matriculado conjuntamente con los pagos realizados.

Se ha selecciona los casos de uso: matricular y pagos, para detallar el prototipo, la secuencia normal de eventos y el diagrama de robustez.

- **Caso de Uso: Matrícular**
  - Prototipo de pantalla para registrar una matrícula

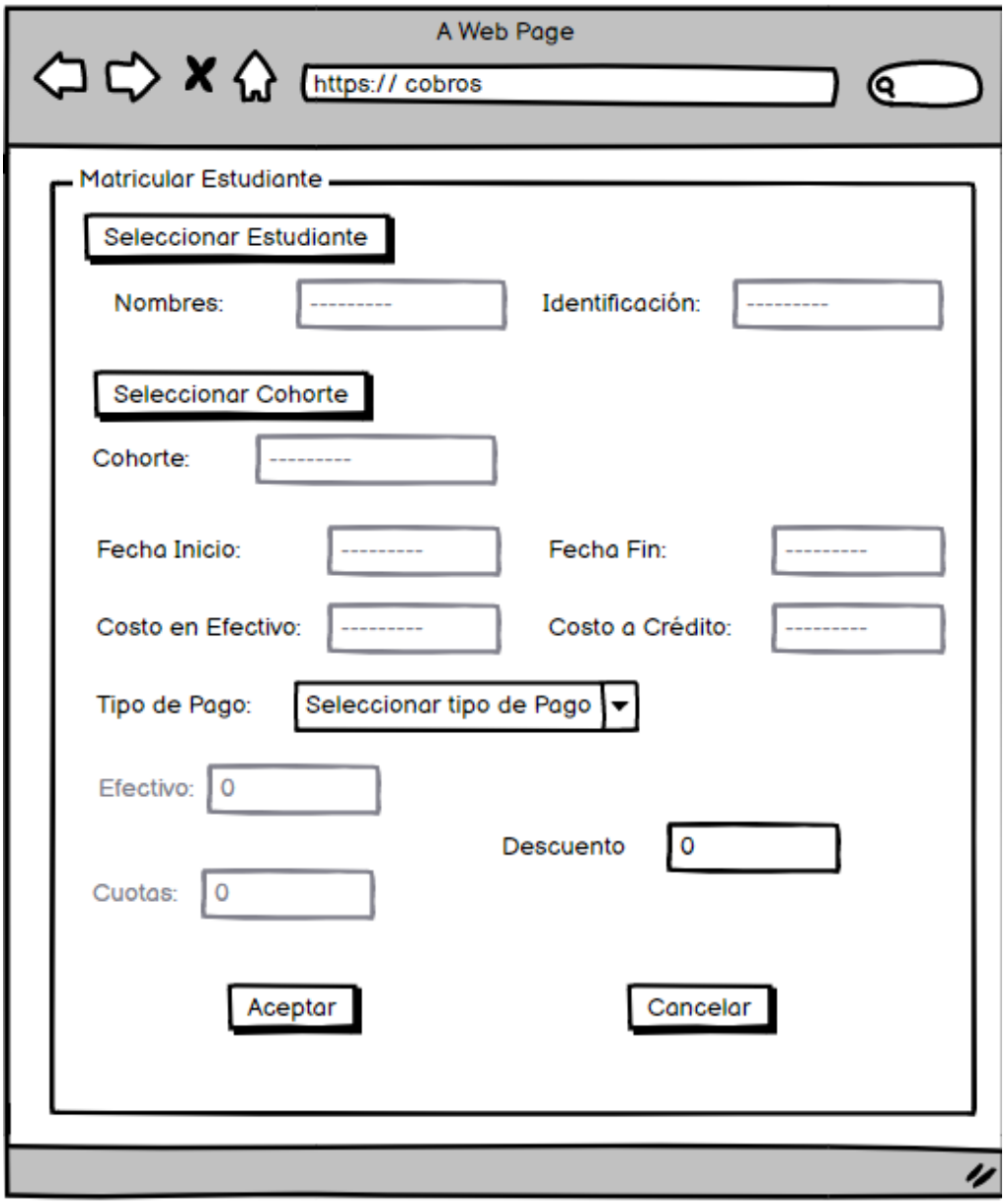


Figura 12 Prototipo de pantalla para matricular un estudiante

- Especificación del caso de uso: Crear Matrícula

Tabla 11 Especificación del caso de uso matricular

Caso de Uso	Crear Matrícula	**Identificador:** CU5
Actores	**Administrador, secretaria**	
Tipo	Primario	
Referencias	RF006	
Precondición	Iniciar Sesión Ingresar en la interfaz de Crear Matrícula	
Secuencia Normal para registrar una matrícula		
Usuario	**Sistema**	
1. Selecciona el botón “Buscar Estudiante” 2. Seleccionar/Buscar el estudiante		
	3. Muestra datos del estudiante seleccionado	
4. Selecciona el botón “Cohorte” 5. Seleccionar/Buscar la Cohorte a la que se va a matricular.		
	6. Muestra los datos de la Cohorte seleccionada	
7. Seleccionar el tipo de pago “Efectivo”		
	8. Activa el campo Efectivo	
9. Ingresa el valor del pago		
10. Ingresar el valor de descuento, en caso de haberlo o si no dejar el valor por defecto		
	11. El sistema valida la información ingresada.	
	12. El sistema guarda los datos	
	13. El sistema muestra un mensaje de confirmación.	
	14. El caso de uso termina.	
Secuencia Alterna		
Usuario	**Sistema**	
7.Selecciona tipo de pago “Crédito”	**7.1.** Activa el campo Cuotas	
1.1. Ingresa cantidad de cuotas Vuelve al paso 9		
Excepciones	**Sistema**	
3. Error al ingresar los datos	a. Muestra un mensaje de error en caso de que algún dato ingresado sea incorrecto. b. Regresar al paso 1, hasta cumplir el flujo.	
Postcondición	Registro de la matricula con éxito.	

- Diagrama de Robustez del proceso para registrar una matrícula

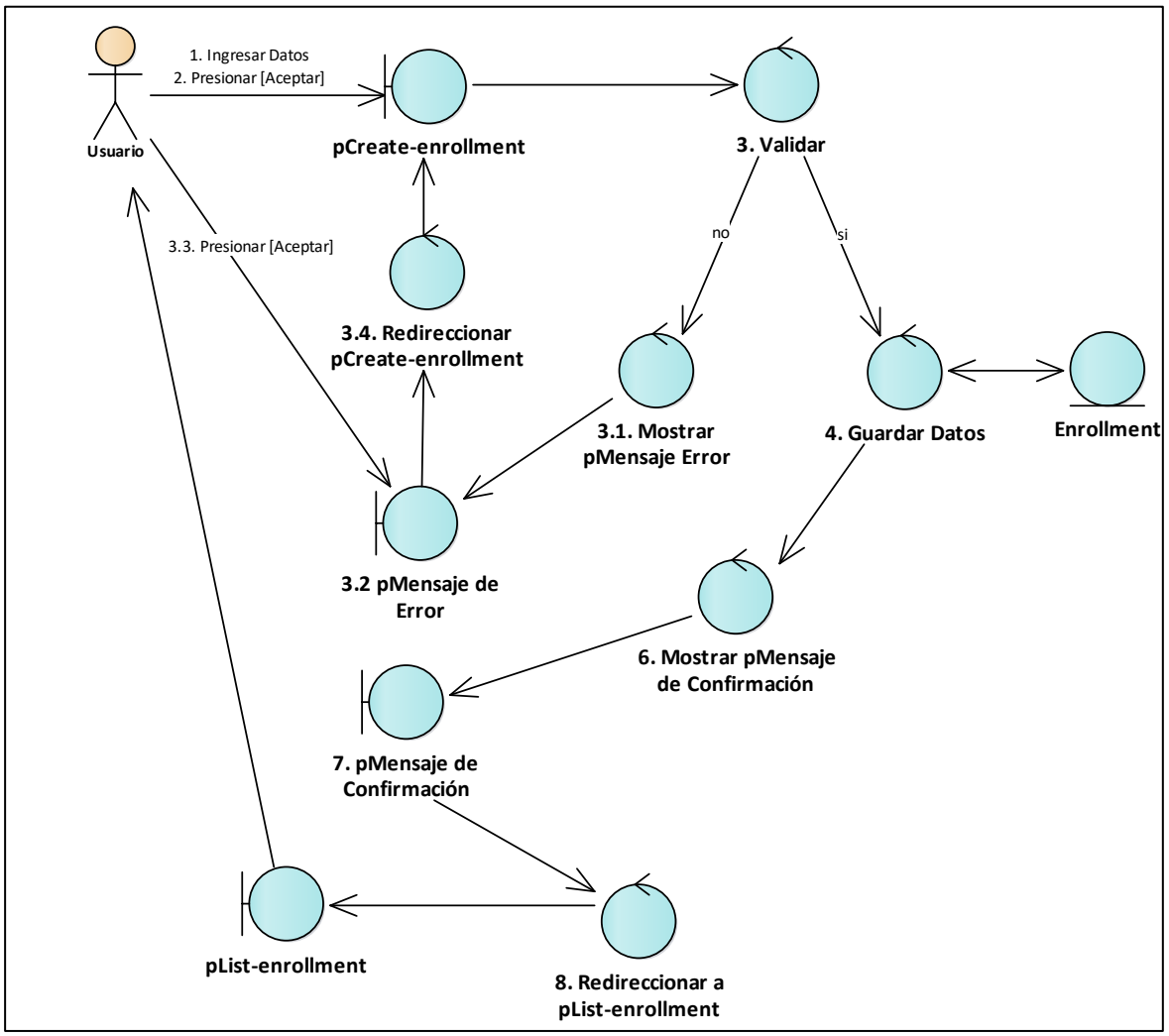


Figura 13 Diagrama de Robustez para registrar una matrícula

- **Caso de Uso: Registrar Pago**

    - Prototipo de pantalla para registrar un pago

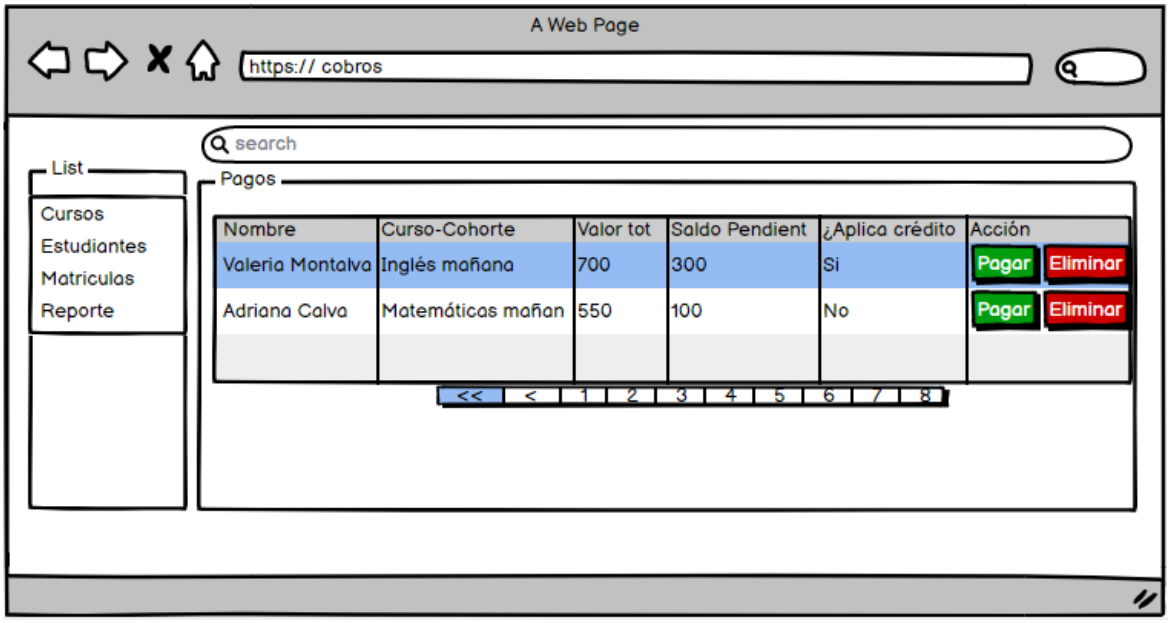


Figura 14 Prototipo de Pantalla para visualizar pagos

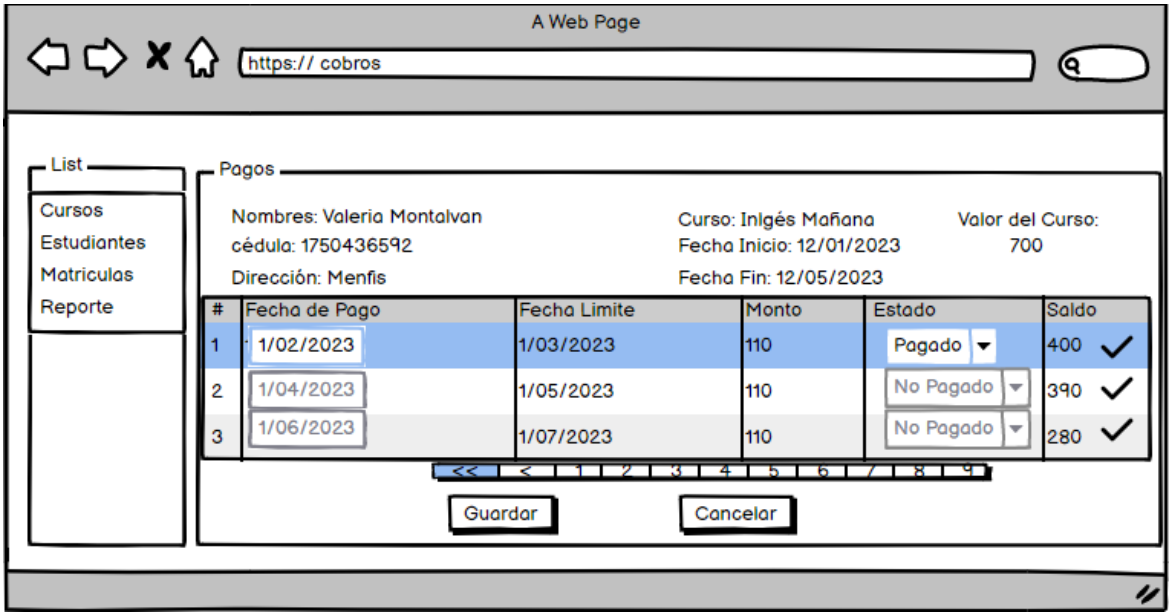


Figura 15 Prototipo de pantalla para registrar un pago

- Especificación del caso de uso: Registrar Pagos

Tabla 12 Descripción del caso de uso Gestionar Pago

Caso de Uso	Registrar Pagos	**Identificador:** CU7
Actores	**Administrador, secretaria**	
Tipo	Primario	
Referencias	RF007	
Precondición	Iniciar Sesión Ingresar a la interfaz Registrar Pagos Al menos un pago registrado	
Secuencia Normal para registrar un pago		
Usuario	**Sistema**	
	1. Busca todos los estudiantes registrados	
	2. Carga la lista de estudiante en la tabla "Pagos", de la interfaz Listar Pagos	
3. Selecciona un estudiante y da clic en el botón "Pagar".		
	4. Muestra la interfaz registrar Pago	
5. Ingresar la fecha del pago 6. Cambiar el estado de pago a "PAGADO" 7. Clic en Guardar		
	8. Valida la acción realizada.	
	9. Guarda los datos	
	10. Muestra un mensaje de confirmación.	
	11. El caso de uso termina.	
Excepciones	**Sistema**	
8. Error al ingresar los datos	a. Muestra un mensaje de error en caso de que algún dato ingresado sea incorrecto. b. Regresar al paso 5, hasta cumplir el flujo.	
Postcondición	Se registra el pago con éxito.	

- Diagrama de Robustez del proceso para registrar un pago

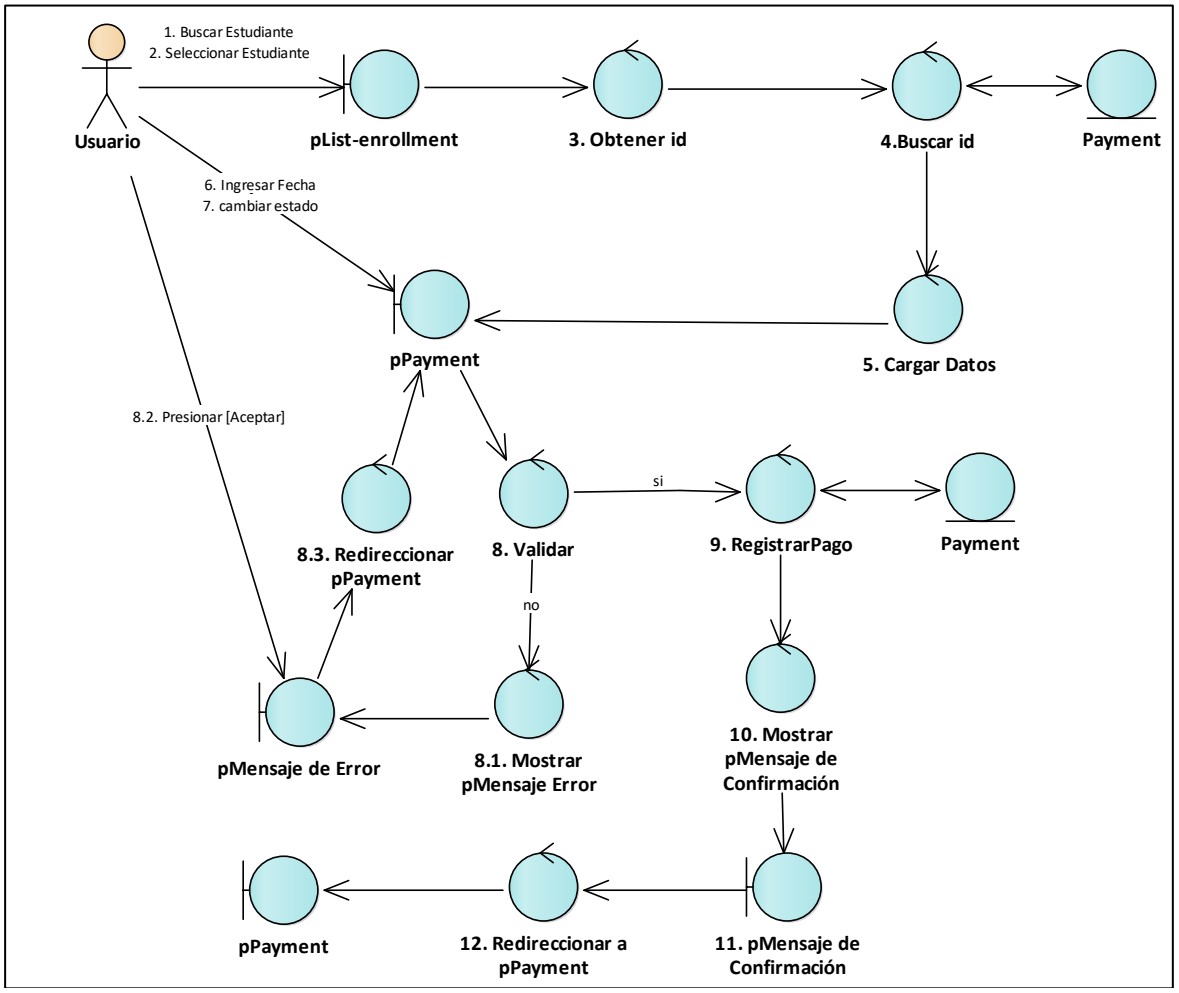


Figura 16 Diagrama de Robustez para registrar un pago

En la figura 16 se muestra el proceso que se cumple para registrar un pago mediante el uso de interfaces, controladores y bases de datos. De esta manera queda plasmada una idea más clara y precisa de cómo funciona el proceso, de qué hacer y cómo se debe hacerlo. Para una mejor comprensión se recomienda revisar el anexo 4 donde se especifica cada uno de los diagramas de robustez.

- **Diagrama del dominio Actualizado**

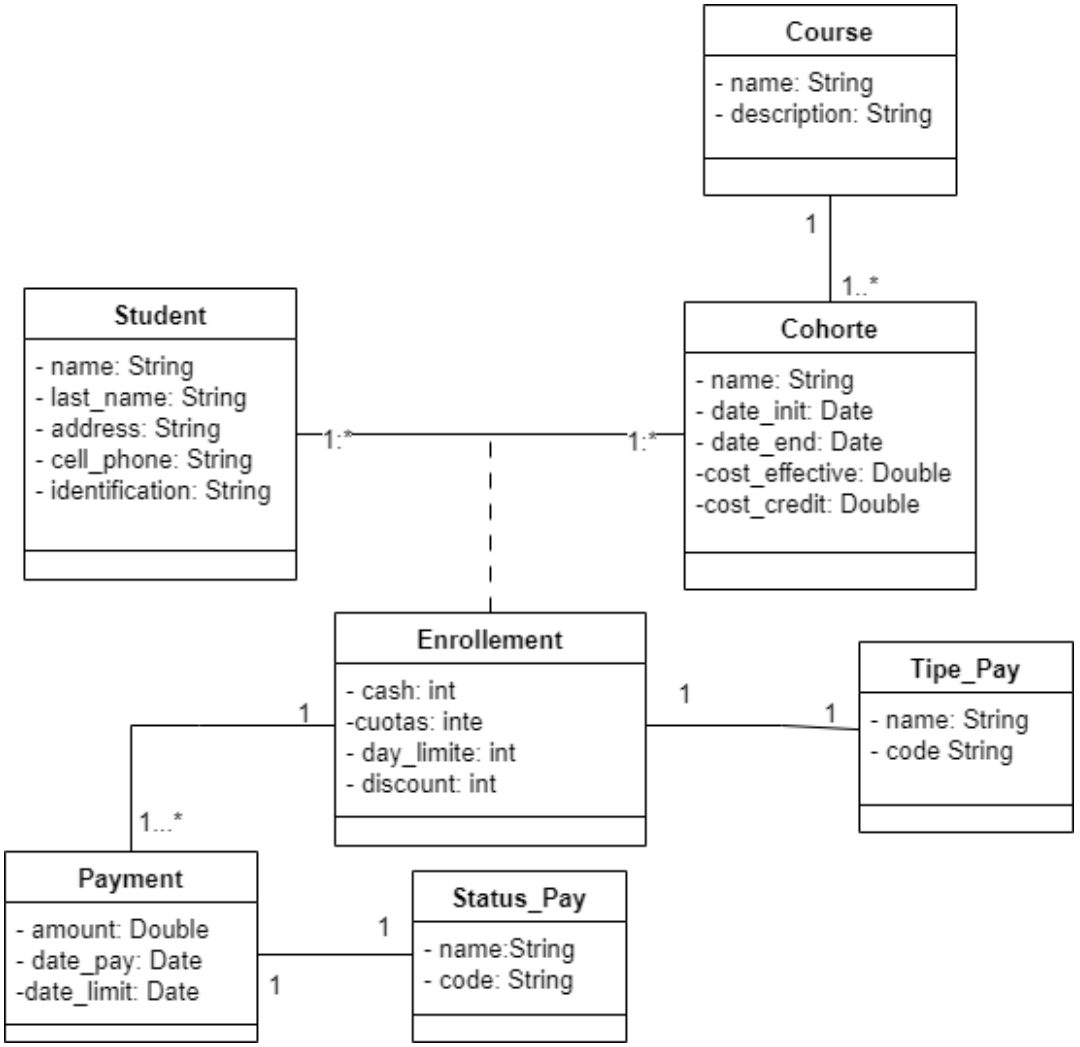


Figura 17 Modelo del dominio actualizado

6.2.1. Diseñar el estilo arquitectónico

En el estilo arquitectónico señalado en la figura 18 de acuerdo a las características de funcionalidad y de no funcionalidad, se ha determinado un tipo de arquitectura Cliente-Servidor y REST. La arquitectura Cliente-Servidor porque el diseño cuenta con un proveedor (Servidor Web) y la parte consumidora (Cliente o Usuarios Finales), los usuarios finales pueden acceder a la información brindada por parte del servidor a través de un computador o una aplicación instalada en su celular, los cuales deben tener conexión a internet. La arquitectura REST se usa con el objetivo de separar el cliente del servidor, al separar la interfaz de usuario del servidor y del almacenamiento de datos, permite a los distintos componentes evolucionar de forma independiente. El servicio contiene la lógica de

negocio el cual proporciona los datos mediante una serie de recursos URL para que el cliente pueda acceder a dichos datos realizará la comunicación utilizando el protocolo HTTP enviando mensajes en formato JSON.

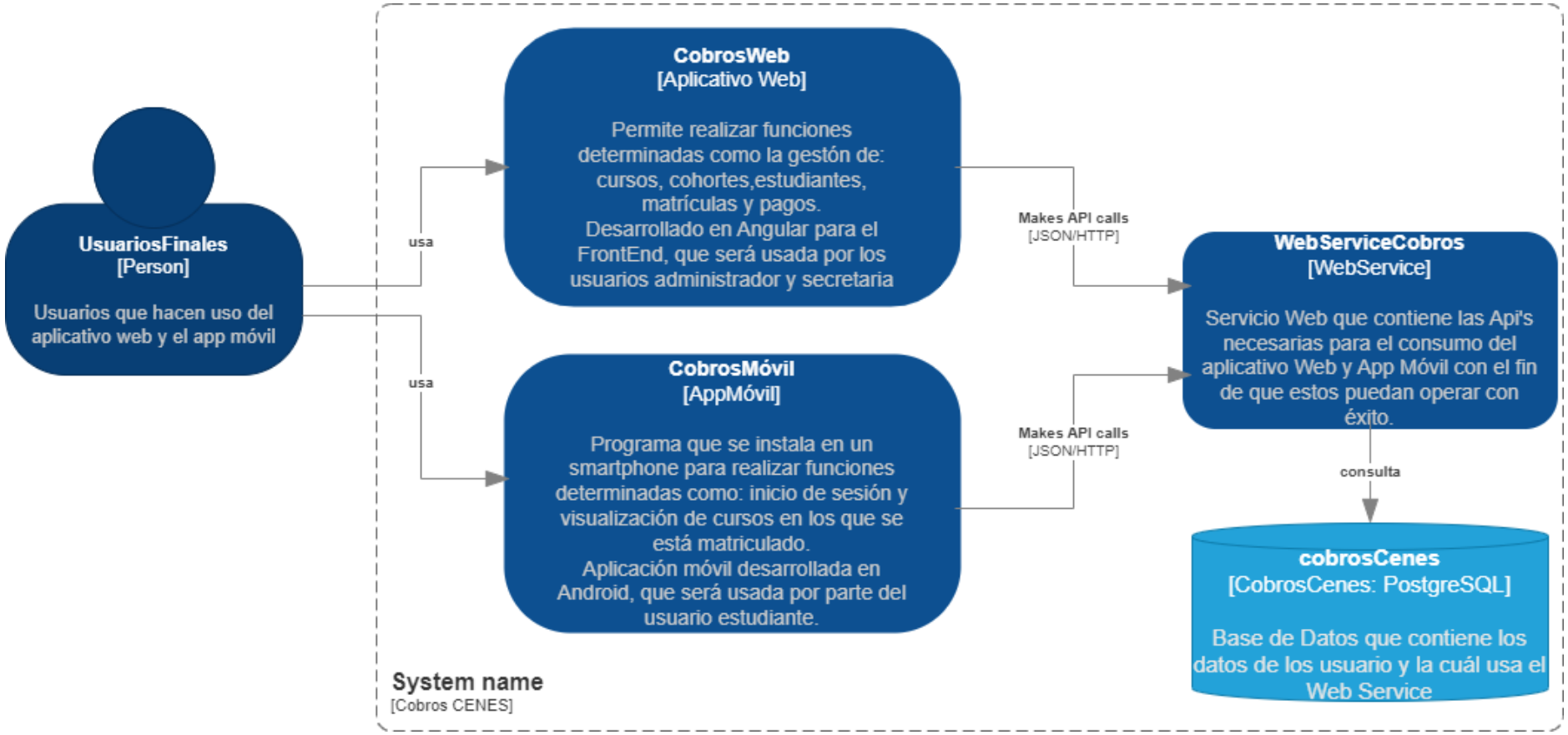


Figura 18 Diseño del estilo arquitectónico

6.2.2. Realizar los diagramas de secuencia

Mediante el uso de los diagramas de secuencia se puede observar de manera clara la línea de vida de un proceso, por esta razón se ha realizado el diagrama de secuencia de cada uno de los casos de uso, de los cuales se ha seleccionado los diagramas de secuencia de matricular y pagos, los que se visualizan en la figura 19 y 20. Para mejor detalle de los diagramas revisar el anexo 5.

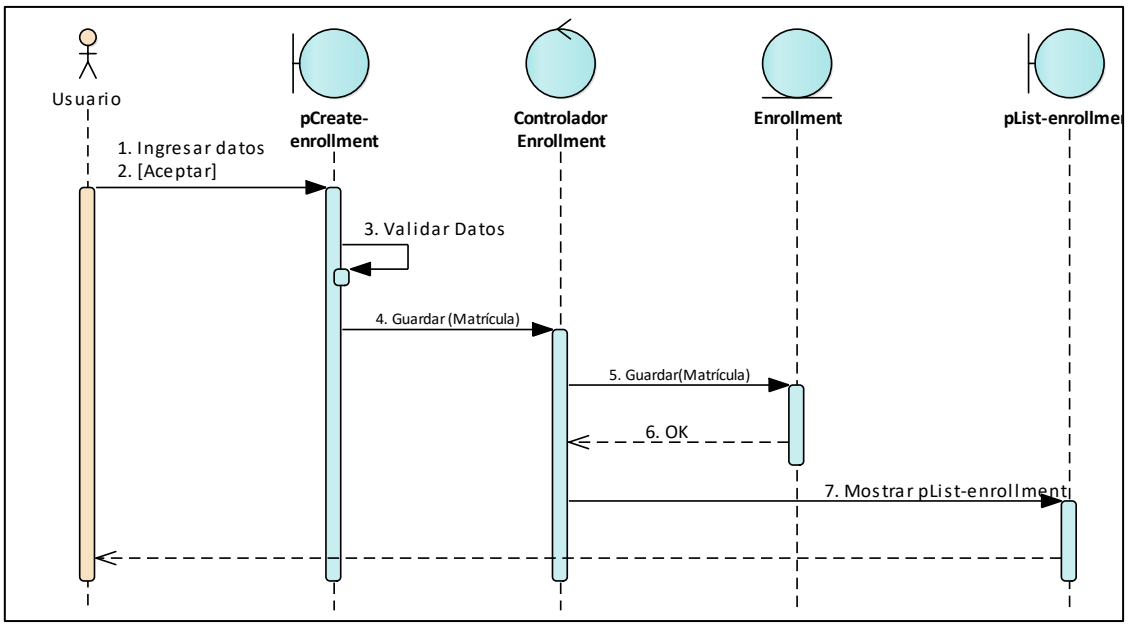


Figura 19 Diagrama de Secuencia para matricular un estudiante

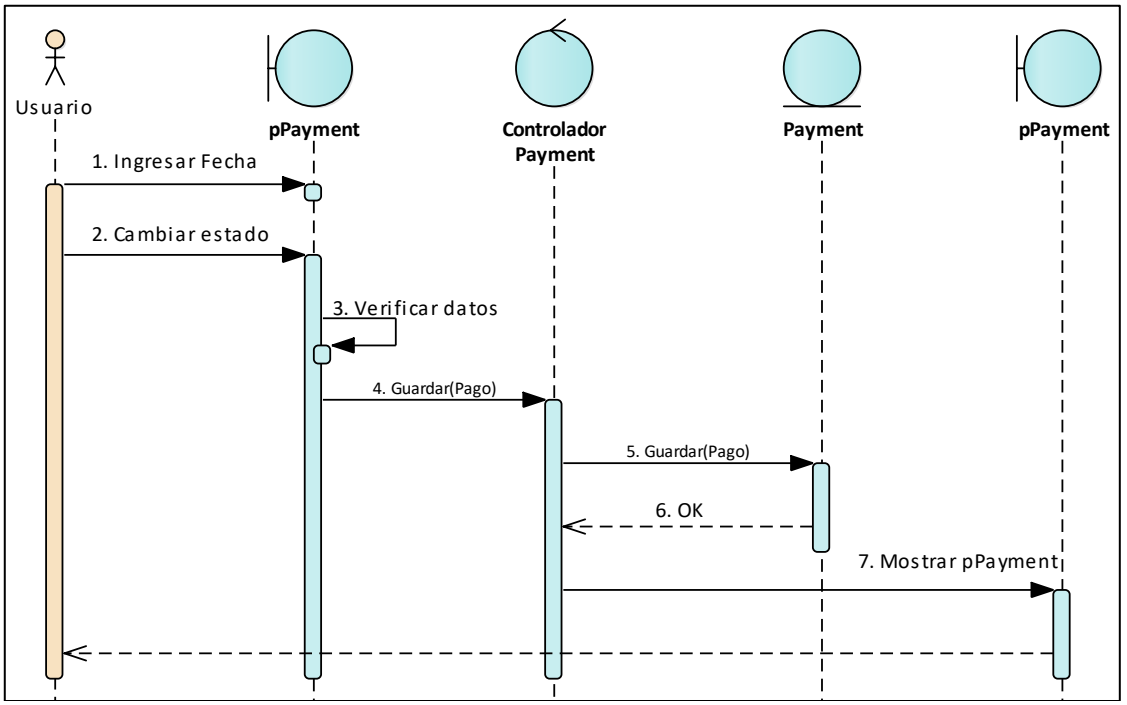


Figura 20 Diagrama de Secuencia para registrar un pago

6.2.3. Generar los artefactos de diseño

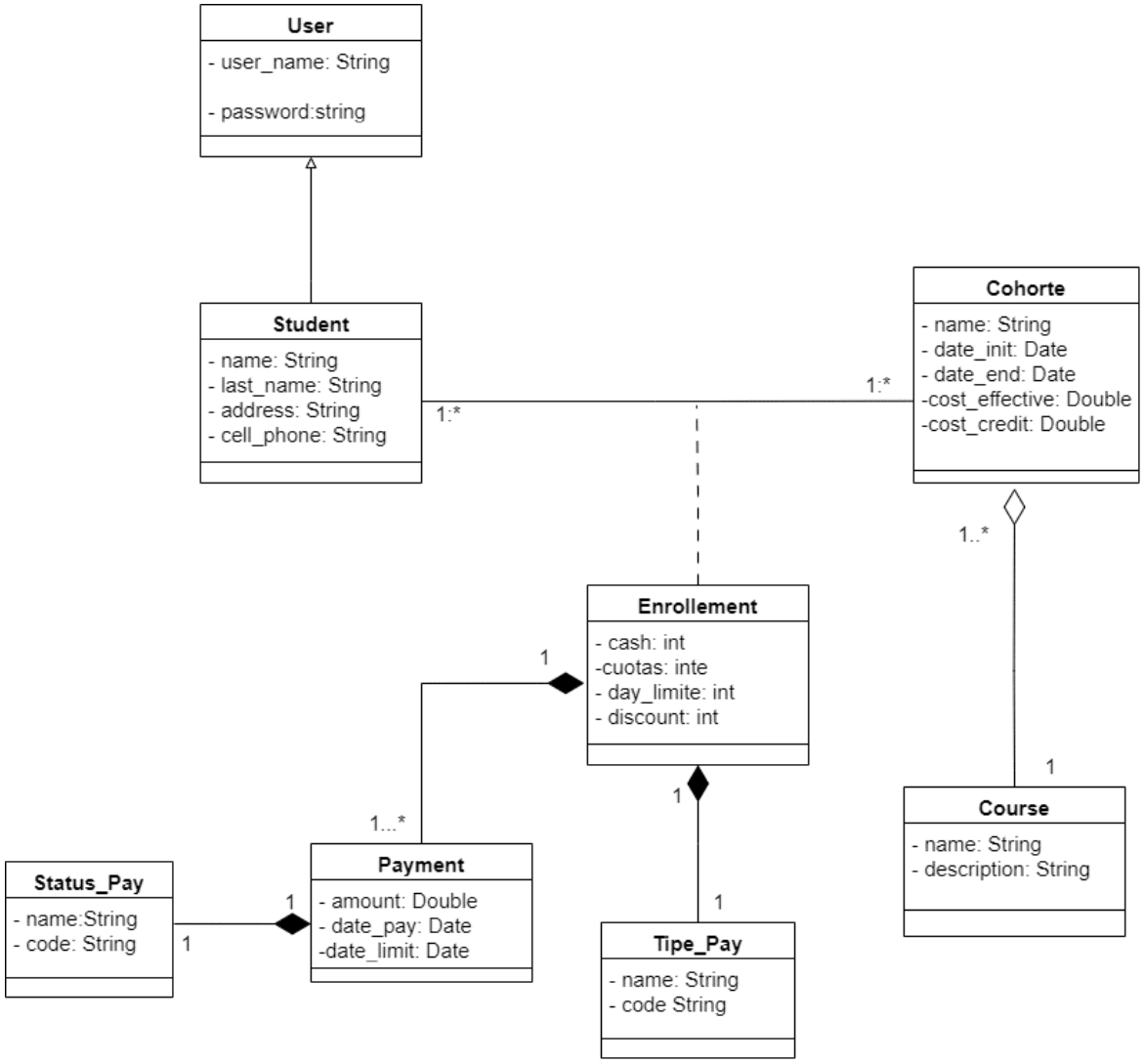


Figura 21 Diagrama de clases

6.2.4. Realizar los diagramas de componente y despliegue

- **Vista de Componentes**

  En el TT los componentes generales, se dividen en 4 componentes básicos: aplicativo web, aplicativo móvil, el servicio web y base de datos. El componente web y móvil van a estar unidos hacia el servicio web por llamadas de api RestFul, enviando datos de tipo JSON, mientras tanto el servicio web realiza consultas al componente de base de datos. En la Figura 22, se puede observar la organización de los diferentes componentes que forman parte del sistema para la gestión de cobros.

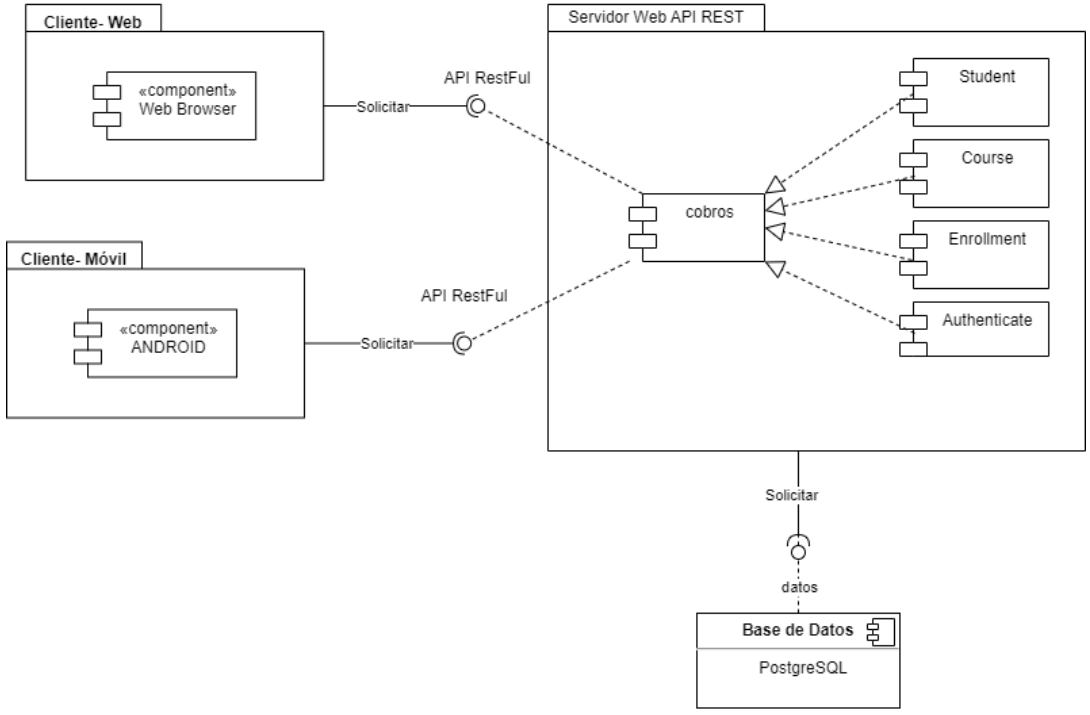


Figura 22 Diagrama de Componentes

- **Vista de Despliegue**

  En la Figura 23 se observa el diagrama de despliegue del sistema para la gestión de cobros, el cual consta de: un equipo para navegador web, un equipo para dispositivo móvil, un servidor web y un servidor de Base de datos, el mismo que no está dentro del servidor web.

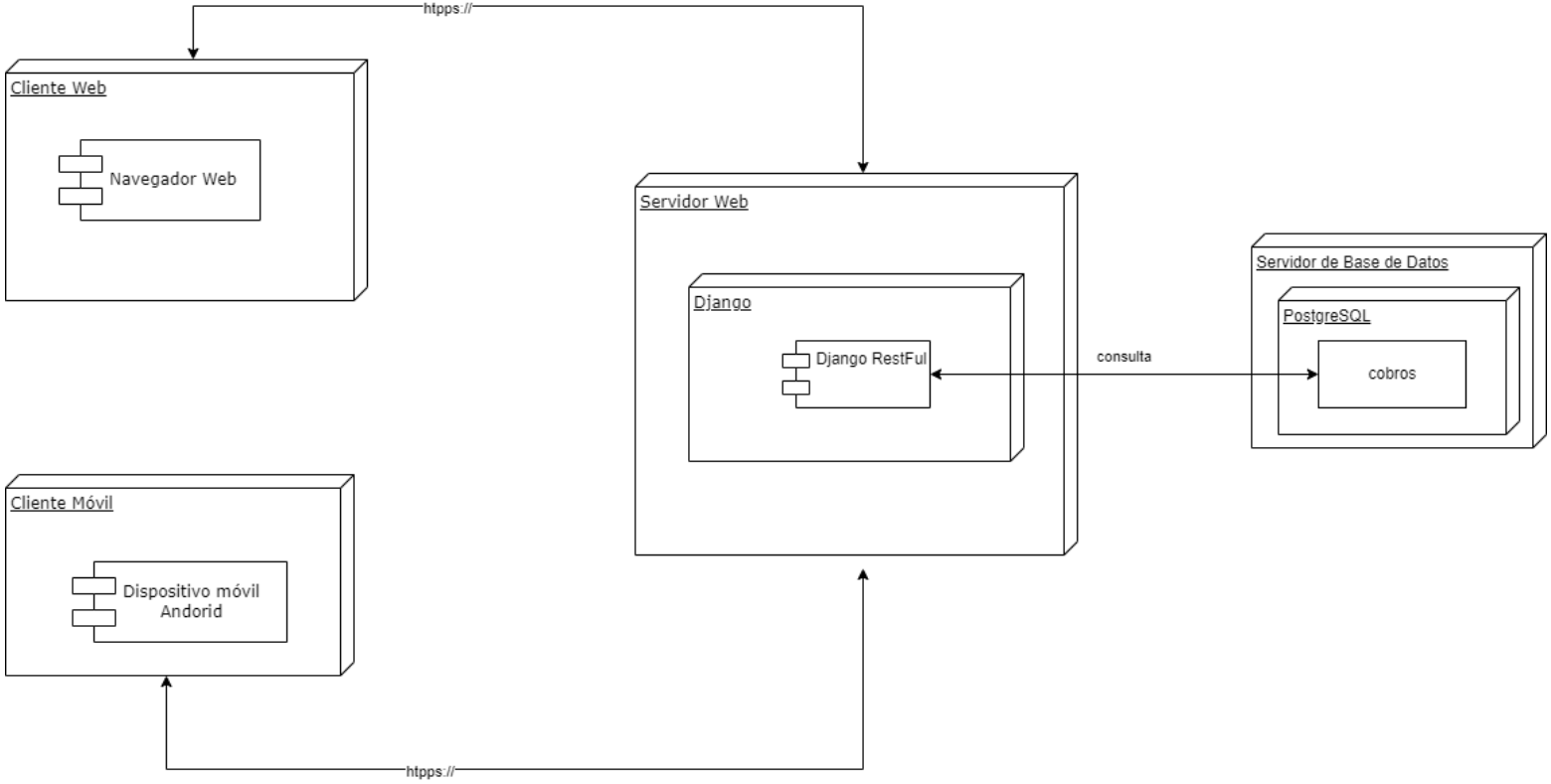


Figura 23 Diagrama de Despliegue

6.2.5. Codificación

Finalizado el diseño arquitectónico, se procedió a la codificación de la solución informática. La estructura que lo compone es: *(i)* el backend que está desarrollado en Django RestFramework, orientado al patrón de diseño Modelo Vista Controlador (MVC), *(ii)* el frontend para el aplicativo web codificado en Angular y *(iii)* el aplicativo móvil se codifico en Flutter.

1. Implementación del backend

El backend se implementó en el framework Django para el servidor de aplicaciones web y se instaló el entorno de Django REST Framework para implementar los métodos del servicio web. La figura 24 muestra la estructura del proyecto API-COBROS.

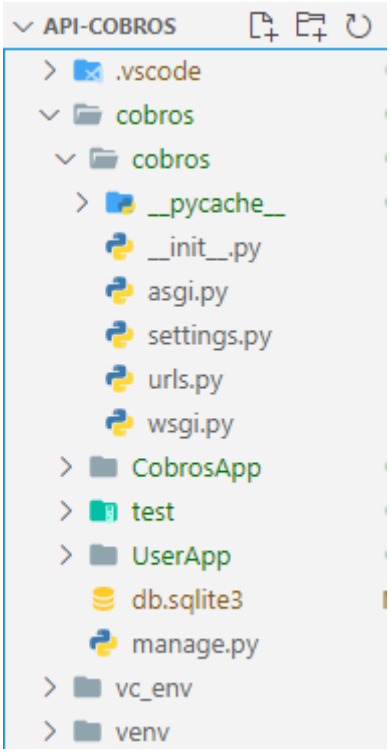


Figura 24 Estructura del servicio web

Entre los directorios más importantes de API-COBROS:

- **url**: son las rutas que permiten acceder a los diferentes recursos de los métodos definidos en el controlador (views) que serán consumidos por el usuario. Además, se encargan de reenviar las solicitudes admitidas hacia el controlador/views respectivo.

  Las rutas definidas en el proyecto son: *(i)* courses, *(ii)* cohortes, *(iii)* students, *(iv)* enrollment, *(v)* payments, *(vi)* tye-pays, *(vii) status-pays*.

En la figura 25 se observa la definición de las rutas, la que redirige la petición hacia el controlador/views pertinente.

```
urlpatterns = [
    path('courses/',CourseAV.as_view(),name='listado-cursos'),
    path('courses/<int:pk>',CourseDetail.as_view(),name='detalle-cursos'),

    path('cohortes/',CohorteAV.as_view(),name='listado-cohortes'),
    path('cohortes/<int:pk>',CohorteDetail.as_view(),name='listado-cohortes'),

    path('students/',StudentAV.as_view(),name='listado-students'),
    path('students/<int:pk>',StudentDetail.as_view(),name='detalle-students'),

    path('enrollements/',EnrollementAV.as_view(),name='listado-matriculas'),
    path('enrollements/<int:pk>',EnrollementDetail.as_view(),name='detalle-matricula'),

    path('payments/',PaymentAV.as_view(),name='listado-pagos'),
    path('payments/enrollement/<int:pk>',getAllByPaymentsEnrrollementId,name='obtener-
todos-pagos-estudainte'),
    path('payments/<int:pk>',PaymentDetail.as_view(),name='detalle-pagos'),

    path('type-pays/',TypePayseAV.as_view(),name='listado-tipo-pagos'),

    path('status-pays/',StatusPaysAV.as_view(),name='listado-estados-pagos'),
]
```

Figura 25 Rutas para redireccionar a las vistas

- **views**: también conocido como controlador, es el lugar donde se controla toda la lógica de la codificación. Se implementan las clases con sus respectivos métodos para consumir los servicios desde el frontend que es el encargado de mostrar los datos al cliente.

  Interactúa con el modelo para obtener los datos solicitado por la ruta y presentarlos a través de la vista. En la figura 26 se presenta el código del controlador/views de pago, en este se procesan los datos de la solicitud y recupera la información necesaria del modelo para que procese la vista y los muestre al cliente.

```
class PaymentAV(APIView):
    def get(self, request):
        data=None
        try:
            enrollement=Payment.objects.all()
            serializer=PaymentSerializer(enrollement,many=True)
            data=serializer.data
            return Response({'data':data,'success':True,'message':'Listado de los pagos'},
                            status=status.HTTP_200_OK)
        except Exception as e:
            return Response({'data':data,'success':False,'message':'Error '+str(e)},
                            status=status.HTTP_404_NOT_FOUND)
    def post(self,request):
        data=None
        try:
            serializer=PaymentSerializer(data=request.data)
            if serializer.is_valid():
                serializer.save()
                data=serializer.data
                return Response({'data':data,'success':True,'message':'Pago  creado exitosamente'},
                                status=status.HTTP_201_CREATED)
            else:
                return Response({'data':serializer.errors,'success':False,'message':'No se puede
crear el pago'},
                                status=status.HTTP_400_BAD_REQUEST)
        except Exception as e:
            return Response({'data':data,'success':False,'message':'Error '+str(e)},
                            status=status.HTTP_404_NOT_FOUND)
```

Figura 26 View Payment

- **models:** Hace referencia a la estructura de la base de datos, es la representación de los datos contenidos en el diagrama de clases. Cada clase hereda de models.Model, se define variables de clase, las que representan los campo de una tabla. En la figura 27 se muestra la definición completa de la clase Pago.

```
class Payment(models.Model):
    amount=models.DecimalField(max_digits=20, decimal_places=2,default=Decimal(0.00))
    date_pay=models.DateField(max_length=250)
    date_limit=models.DateField(max_length=250)
    status_pay=models.ForeignKey(Status_Pay,on_delete=models.RESTRICT,
                                 related_name='statuslist')
    enrollement=models.ForeignKey(Enrollement,on_delete=models.RESTRICT,
                                  related_name='enrrollementlist')
    updated_on=models.DateTimeField(auto_now=True)
    created_on=models.DateTimeField(auto_now_add=True)
```

Figura 27 Modelo de Payment

- **serializer:** toma la estructura de un modelo para convertir su estructura en un formato JSON, en la figura 27 se muestra el serializer de Payment donde se establece cuáles serán los parámetros de respuestas que devolverá la view, de la misma manera permite la verificación de que los datos de ingreso sean los que se pide dentro de la clase Payment. Además, cuenta con una función para crear un pago, en la cual se establece los datos que se deben recibir.

```
class PaymentSerializer(serializers.ModelSerializer):
    #user=UserSerializer(read_only=True)
    status_pay = StatusPaySerializer(read_only=True)
    status_pay_id=serializers.SlugRelatedField(queryset=Status_Pay.objects.all(),
                                               slug_field='id', write_only=True)
    enrollement = EnrollementSerializer(read_only=True)
    enrollement_id=serializers.SlugRelatedField(queryset=Enrollement.objects.all(),
                                                slug_field='id', write_only=True)
    class Meta:
        model = Payment
        fields = [  'id',
                    'amount',
                    'date pay',
                    'date_limit',
                    'status_pay',
                    'status_pay_id',
                    'enrollement',
                    'enrollement_id'
                ]
    def create(self, validated_data):
        data = {
            'amount': validated_data.get('amount', None),
            'date_pay': validated_data.get('date_pay', None),
            'date_limit': validated_data.get('date_limit', None),
            'status_pay': validated_data.get('status_pay_id', None),
            'enrollement': validated_data.get('enrollement_id', None)
            }
        return Payment.objects.create(**data)
    def update(self,instancia,validated_data):
        instancia.amount=validated_data.get('amount',instancia.amount)
        instancia.date_pay=validated_data.get('date_pay',instancia.date_pay)
        instancia.date_limit=validated_data.get('date_limit',instancia.date_limit)
        instancia.status_pay=validated_data.get('status_pay_id',instancia.status_pay_id)
     instancia.enrollement=validated_data.get('enrollement_id',instancia.enrollement_id)
        instancia.save()
```

Figura 28 Serializer Payment

El patrón MVC, se define en tres partes principales:

- **Modelo:** representa la estructura lógica de los datos de la codificación
- **Vista:** es la presentación al usuario la información contenida en el modelo, consiste generalmente en pantallas.

- **Controlador:** se encarga de aceptar solicitudes que hace el usuario a través del navegador, contacta al modelo para solicitar los datos que necesite y luego toma la vista adecuada para mostrar los datos a los usuarios.

En la figura 29 se visualiza el funcionamiento del patrón MVC, el patrón se aplica en el desarrollo web y móvil.

1. El cliente/ usuario realiza una solicitud mediante una ruta que ejecuta una petición (request) al controlador.
2. El controlador procesa la petición que hizo el usuario y solicita los datos al modelo.
3. El modelo se comunica con la base de datos mediante una consulta y esta retorna la información.
4. El modelo retorna los datos al controlador.
5. El controlador se encarga de retornar la vista al cliente con los datos solicitados que serán presentados al cliente.

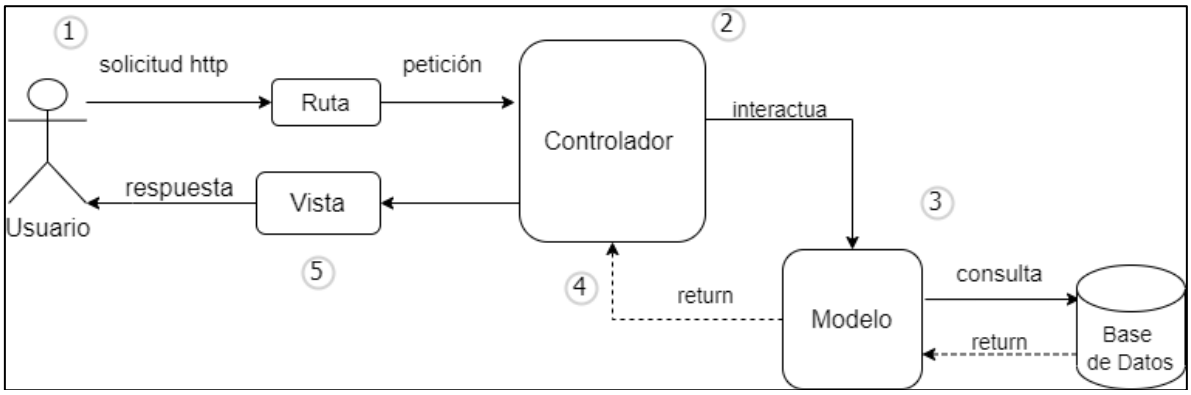


Figura 29 Patrón de Diseño

2. Implementación del frontend web

El frontend se implementó en el framework Angular para la interfaz de la aplicación que se presentará al usuario final. La figura 30 detalla la estructura del proyecto FRONTEND-COBROS.

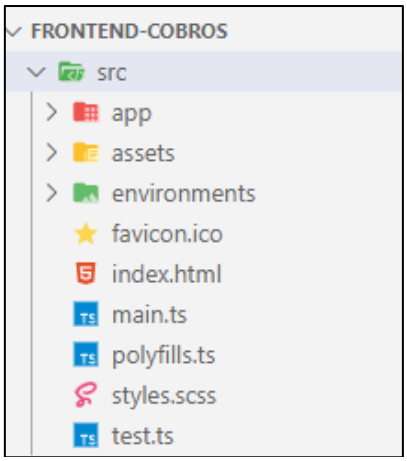


Figura 30 Estructura del frontend

Entre los directorios más importantes de FRONTEND -COBROS, está el directorio Src, que contiene las capetas detalladas a continuación:

- **App:** directorio donde se ubica toda la implementación de los componentes, es decir el código fuente. En la figura 31 se visualiza el directorio app que contiene la carpeta admin, en la que se crearon carpetas para cada uno de los componentes del backend.

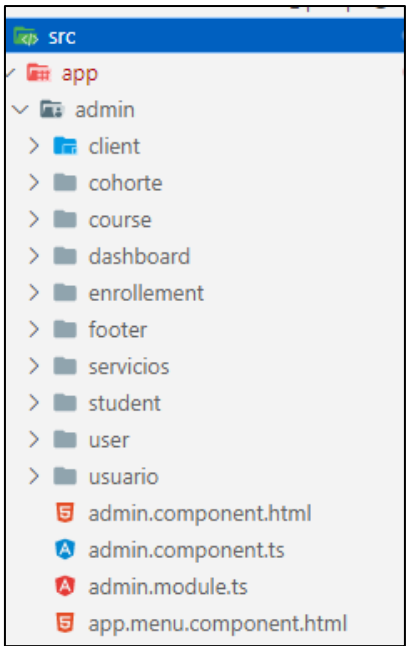


Figura 31 Estructura del directorio app

Cada carpeta contiene archivos html que sirven para crear las vistas finales, la figura 32 detalla el componente para crear una matrícula, para la cual se necesita del modelo *(i)* de cohorte, *(ii)* tipo de pago y *(iii)* estudiante.

```
<div class="card ui-fluid">
    <p-progressBar mode="indeterminate"
*ngIf="!listUsers"></p-progressBar>
    <app-form-enrollement
        (onSubmitCohorte)="createStudent($event)"
        [modeForm]="'Crear Enrollement'"
        [modelCohorteFull]="listCohorte"
        [modelStudentFull]="listUsers"
        [modelTypePaysFull]="listTypePays"
        *ngIf="listUsers">
    </app-form-enrollement>
```

Figura 32 create-enrollment.component.html

La figura 33 presenta el app-routing.module, archivo que contiene todas las rutas o url que unen el código del backend con las vistas .html del frontend.

```
@NgModule({
    imports: [
        RouterModule.forRoot([
            { path: '', redirectTo: '/auth/login', pathMatch: 'full' },
            { path: "home",component:HomeComponent },
            { path: "principal",component:PrincipalComponent },
            { path: "admin",component:AdminComponent,children:[
                { path: 'usuario', loadChildren: () =>
                import('./admin/usuario/usuario.module').then(m => m.UsuarioModule) },
                { path: 'user', loadChildren: () =>
                import('./admin/user/user.module').then(m => m.UserModule) },
```

Figura 33 Rutas para unir backend con frontend

- **Assets:** directorio para imágenes y archivos adicionales que la app necesite. La figura 34 presenta la estructura de la carpeta assets, esta contiene las imágenes que se asignarán a cada componente de vista para mostrar al usuario final.

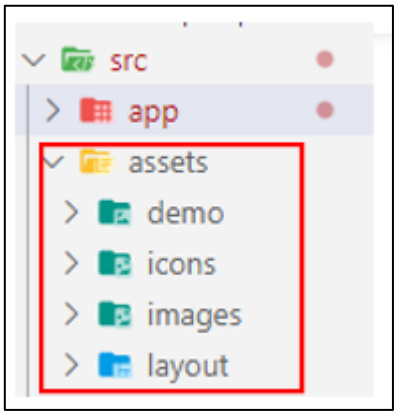


Figura 34 Estructura de assets

- **Enviroments:** directorio de ambientes, contiene el ambiente de desarrollo y producción.

Figura 35 Ambiente para conectarse con el backend

- **index.html**: Archivo HTML principal desde donde se construye toda la aplicación. En la figura 36 se muestra la estructura del archivo index, apartado donde se agrega el nombre e imagen de la página de la aplicación de interfaz principal.

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>CENES COBROS</title>
    <base href="/">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" type="image/x-icon" href="favicon.ico">
    <link id="theme-css" rel="stylesheet" type="text/css"
href="assets/layout/styles/theme/lara-light-indigo/theme.css">
    <script type="text/javascript"
src="https://maps.google.com/maps/api/js?sensor=false"></script>
</head>
```

Figura 36 Página Principal index.html

3. Implementación del frontend móvil

En cuanto al frontend para la aplicación móvil se implementó el SDK Flutter, para desarrollar la interfaz de usuario para un cliente móvil. La estructura de COBROS móvil se detalla en la figura 37.

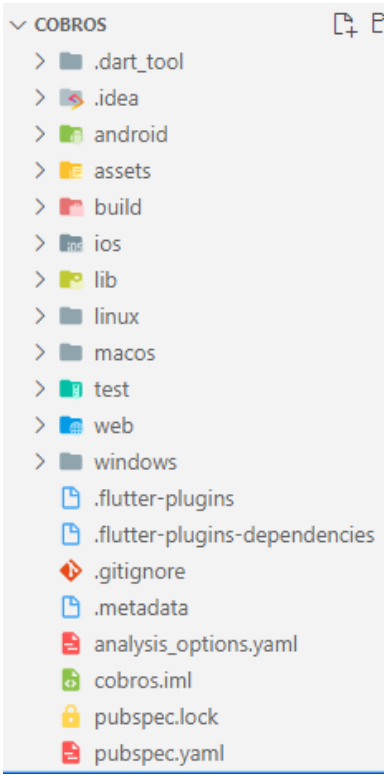


Figura 37 Estructura del proyecto en flutter

Entre los directorios más principales de COBROS esta la carpeta lib, la que cuenta con el archivo de inicio main y 4 carpetas mostradas en la figura 38.

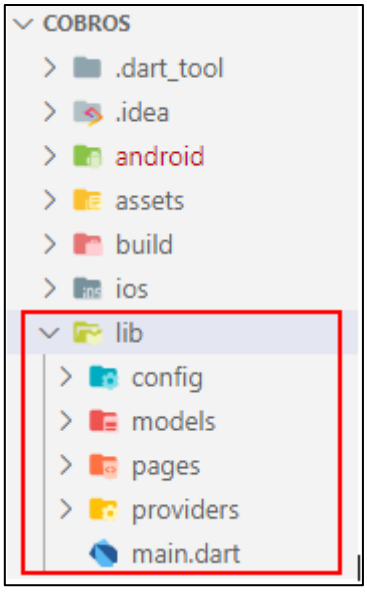


Figura 38 Esquema de la carpeta lib

Config: En este directorio está el archivo utils.dart, en el cual se estableció la variable de la URL base a utilizar para realizar las peticiones, es decir el enlace para comunicarnos con el aplicativo web. La figura 39 detalla el contenido del archivo.

```dart
import 'package:flutter/material.dart';

String URLBASE = 'https://proeditslub.com/cobros';
```

Figura 39 utils.dart

Models: contiene los archivos con los modelos de datos utilizados, *(i)* enrollment, *(ii)* estudiante y *(iii) usuario.* La figura 40 detalla la clase Enrollment con sus respectivos campos.

```dart
class Enrollement {
    Enrollement({
        this.id,
        this.student,
        this.cohorte,
        this.tipePay,
        this.cuotas,
        this.dayLimite,
        this.cash,
        this.discount,
        this.createdOn,
    });
    int? id;
    Student? student;
    Cohorte? cohorte;
    Pay? tipePay;
    int? cuotas;
    int? dayLimite;
    int? cash;
```

Figura 40 paymentEnrollmentmodel.dart

Pages: contiene todas las páginas o interfaces a mostrar en el aplicativo móvil, se ha tomado como ejemplo la página de list.enrollment.page, expuesta en la figura 39. Se encarga de verificar que exista un token y se controla al momento de que el usuario realice scroll y llegue a la parte final se llame al método getNewEnrollments con el fin de mostrar nuevos registros.

```dart
@override
 void initState() {
   super.initState();
   verifyTokenPreference();

   _scrollController = ScrollController();
   getNewEnrollements();

   _scrollController.addListener(() {
     if (_scrollController.position.pixels ==
         (_scrollController.position.maxScrollExtent)) {
       getNewEnrollements();
     }
```

Figura 41 list.enrollment.page

4. Diseño final de interfaces de Usuario del aplicativo web para gestión de cobros

A continuación, se muestra las pantallas finales del aplicativo web CENES-COBROS.

En la figura 42 se puede visualizar la pantalla para iniciar sesión.

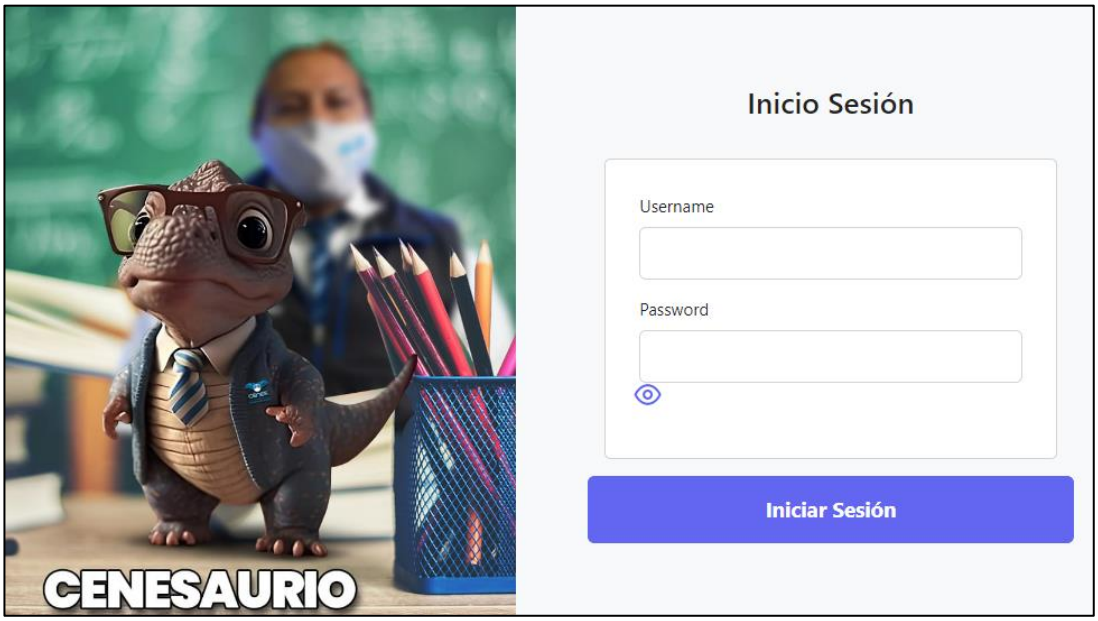


Figura 42 Pantalla fin de Inicio Sesión

Iniciada sesión, se muestra la pantalla de inicio tal como se visualiza en la figura 43.

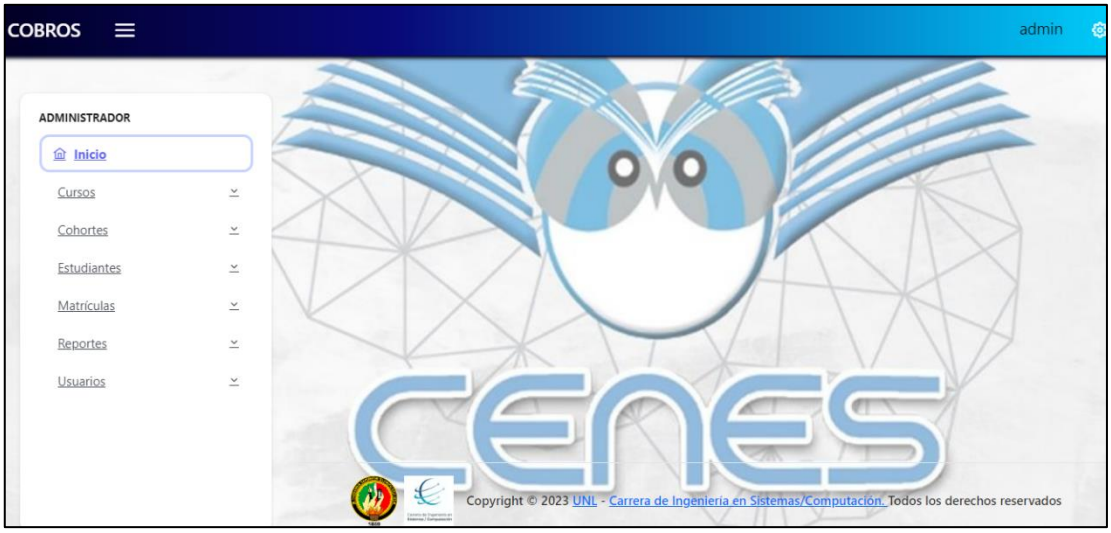


Figura 43 Pantalla de inicio del aplicativo web

En la figura 44 se puede observar la pantalla de matrículas y en la figura 45 la pantalla para registrar un pago.

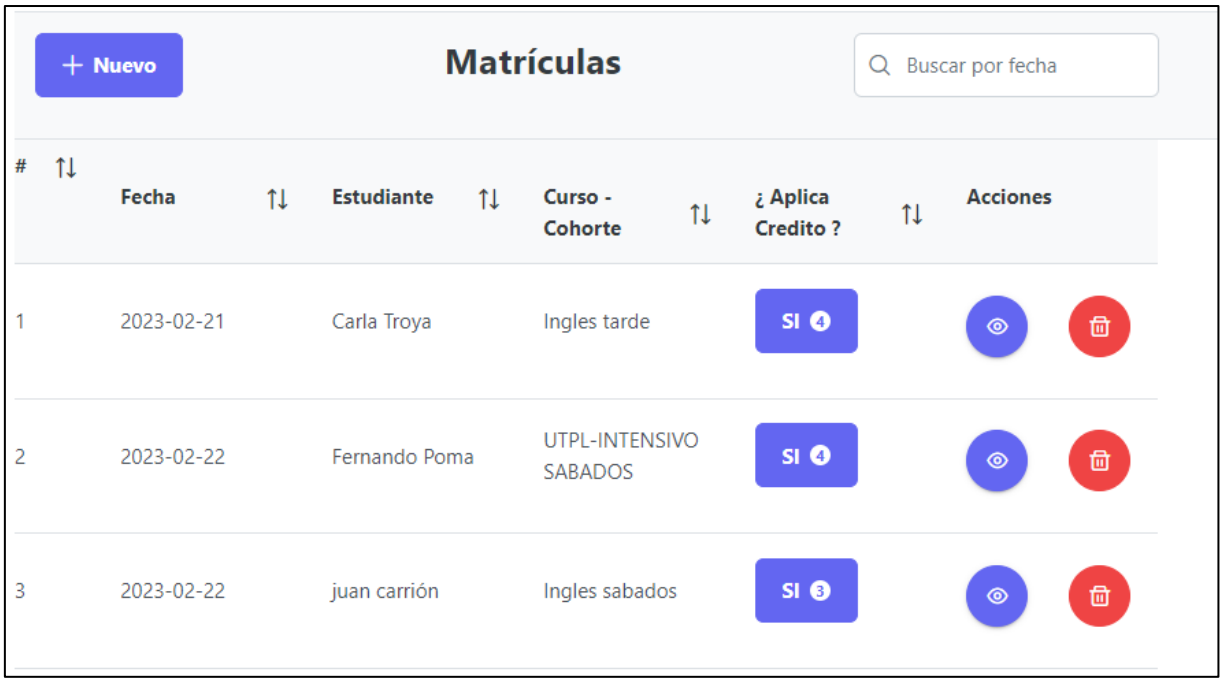


Figura 44 Pantalla de matrículas del aplicativo web

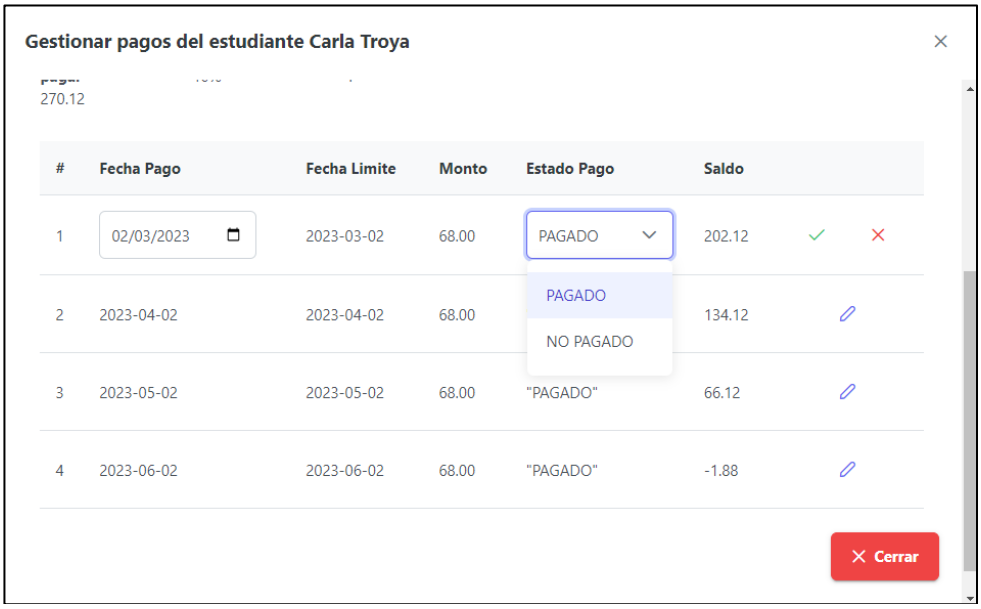


Figura 45 Pantalla para registrar un pago en el aplicativo web

Para concluir, el código de todos los componentes de software se encuentra almacenado u organizado en el repositorio de GitHub, en las direcciones:

- Para el backend: https://github.com/CarlaTroy/API-COBROS.git
- Para el frontend web: https://github.com/CarlaTroy/FRONTEND-COBROS.git
- Para el frontend móvil: https://github.com/CarlaTroy/APP-COBROS.git

Además, el aplicativo web se encuentra en un servidor real con la siguiente dirección: https://proeditsclub.com/cobros/frontend/index.html#/auth/login

6.3. OBJETIVO 3: Evaluar el aplicativo web en un ambiente simulado.

Para realizar la evaluación del aplicativo, se planifico cuatro tipos de pruebas para evaluar la calidad y funcionalidad de todos los artefactos de software, tales pruebas fueron: *(i)* pruebas de caja negra, *(ii)* pruebas de caja blanca, *(ii)* pruebas funcionales de aceptación y *(iv)* pruebas de carga y estrés. Dichas pruebas están descritas a continuación.

6.3.1. Pruebas Caja Negra

En las pruebas de caja negra el principal objetivo es evaluar los datos que se ingresan, el procesamiento y que resultados son los esperados.

Las pruebas tienen dos fases: *(i)* se realiza la evaluación de cada método y *(ii)* se registra en la tabla de casos de prueba si la evaluación cumple o no con el objetivo.

Se realizó la planificación de los métodos a evaluar utilizando la herramienta de Postman, para realizar la evaluación se requiere *(i)* dirección del recurso, *(ii)* método de evaluación y *(iii)* los parámetros a enviar. Además, por cada método evaluado se fue registrando el éxito o fracaso, en la tabla 15 de casos de prueba se puede observar dicho registro.

La prueba fue realizada y dirigida por la tesista, considerando cada uno de los requerimientos establecidos por el cliente, efectuado en dos partes: *(i)* en la tabla 13 y 14 que se eligieron como ejemplo, se detalla el procedimiento de ejecución de la prueba y *(ii)* en la tabla 15 se registra si se cumplió o no de la prueba, en caso de que falle la prueba se agrega el comentario respectivo del fallo. Posteriormente en la tabla 16 se detallan los errores encontrados en las pruebas y las soluciones que se dieron a cada uno. Para mayor detalle de las pruebas de caja negra, revisar el anexo 6.

Tabla 13 Petición para crear una matrícula

URL	Método	Tiempo de ejecución	Descripción
http://127.0.0.1:8000/api/enrollements/ https://proeditsclub.com/cobros/api/enrollements/	POST	36 ms	Permite la creación de una matrícula
RESULTADO			

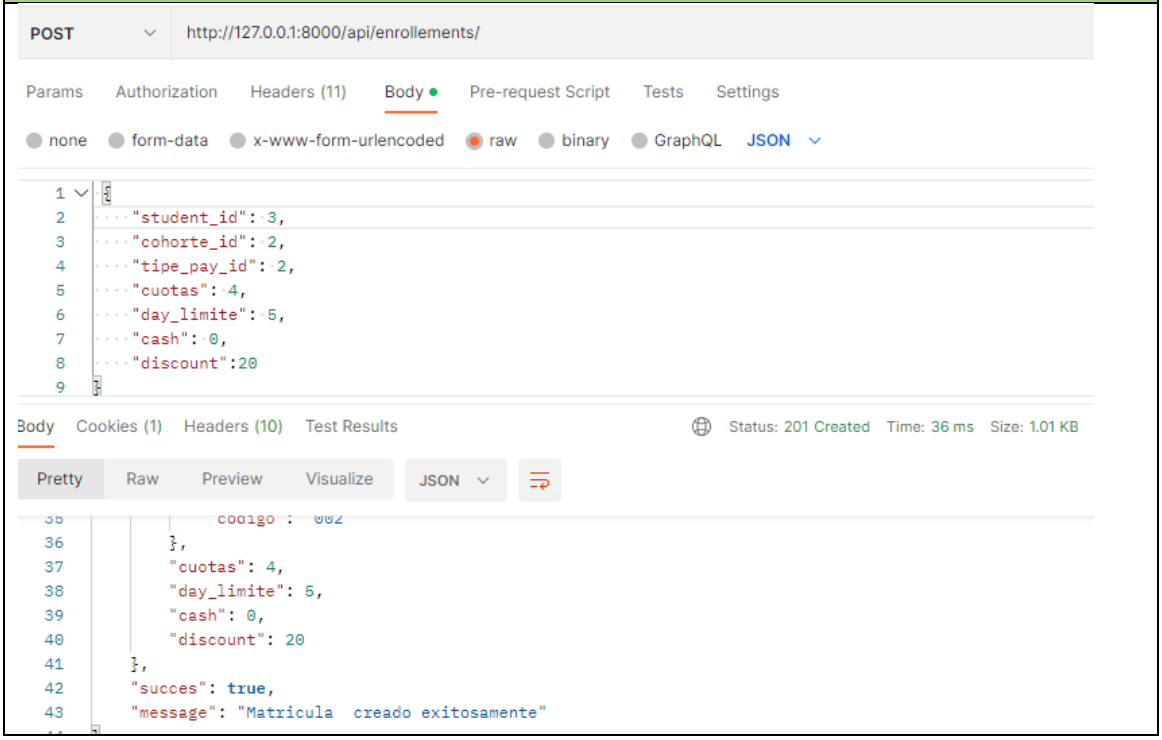
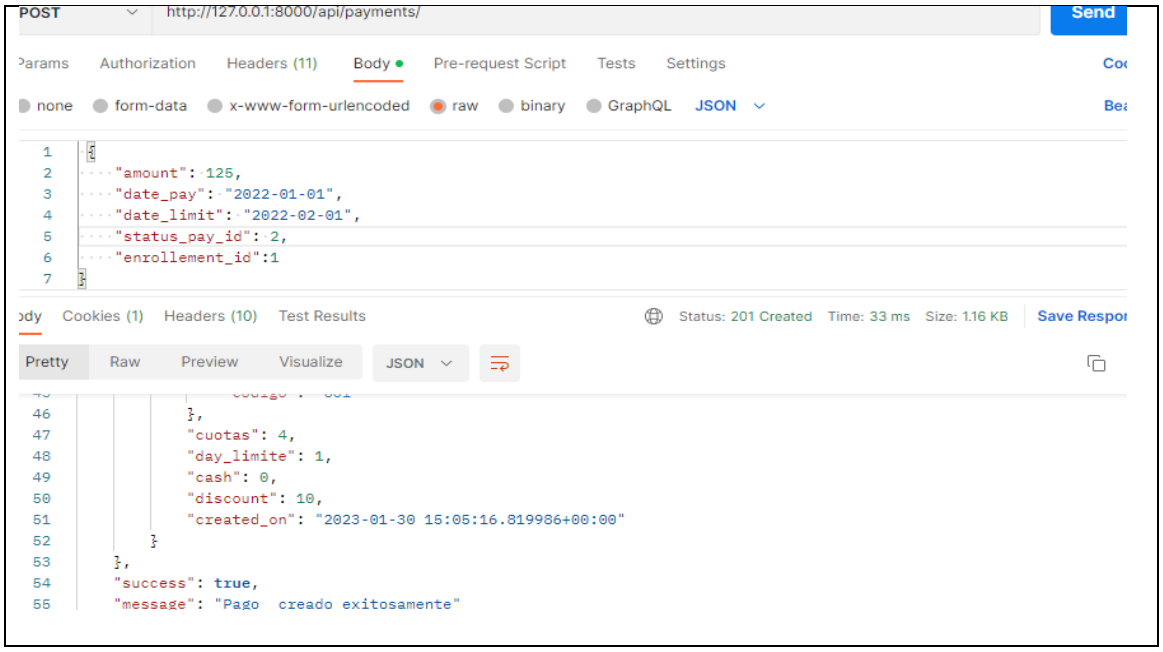


Tabla 14 Petición para registrar un pago

URL	Método	Tiempo de ejecución	Descripción
http://127.0.0.1:8000/api/payments/ https://proeditsclub.com/cobros/api/payments	POST	32 ms	Permite registrar un pago en base a una matrícula creada
RESULTADO			



A continuación, se muestra en la tabla 15, los criterios que se tomaron en cuenta para comprobar cada método en la prueba de caja negra.

Tabla 15 Casos de prueba de Caja Negra de la gestión de cobros

Casos de Prueba de Caja Negra			
Criterio	**CUMPLE**		**Observación**
	Si	**No**	
Iniciar sesión y mostrar mensaje de éxito o motivo del error.	X		
Crear un curso y mostrar mensaje de éxito o motivo del error	X		
Editar un curso y mostrar mensaje de éxito o motivo del error	X		
Eliminar un curso y mostrar mensaje de éxito o motivo del error	X		
Buscar un curso	X		
Crear una cohorte, a partir de un curso y mostrar mensaje de éxito o motivo del error.		X	Error al crear una cohorte
Editar una cohorte y mostrar mensaje de éxito o motivo del error.	X		
Eliminar una cohorte, siempre y cuando no haya estudiantes matriculados, mostrar mensaje de éxito o motivo del error.	X		
Buscar una Cohorte	X		
Crear un Estudiante y mostrar mensaje de éxito o motivo del error.	X		
Editar un Estudiante y mostrar mensaje de éxito o motivo del error.	X		
Buscar un Estudiante.	X		

Crear una matrícula y mostrar mensaje de éxito o motivo del error.	X		
Eliminar matrícula y mostrar mensaje de éxito o motivo del error.		X	Error al eliminar matricula ya que está relacionada con los pagos
Registrar un pago y mostrar mensaje de éxito o motivo del error.	X		
Editar un pago y mostrar mensaje de éxito o motivo del error.	X		

Se recalca que los métodos *(i)* crear cohorte y *(ii)* eliminar matrícula no cumplieron con el criterio de aceptación al momento de realizar la prueba de caja negra. Se identificó los siguientes errores y se dio la solución siguiente:

Tabla 16 Errores y soluciones de las Pruebas de Caja Negra

ERRORES	SOLUCIÓN
Error al eliminar una matrícula. Este error se dio debido a que en el modelo de enrollment, la llave ForeignKey de Payment se encontraba en RESTRICTED, lo que no permitía que se elimine una matrícula.	Al identificar el error se procedió a realizar el cambio en cuanto al valor de llave ForeignKey de Payment contenida en el modelo enrollment, el nuevo valor del ForeignKey es CASCADE, lo que permite que se elimine la matricula conjuntamente con los pagos, que es lo requerido por el cliente.
Error al crear cohorte. Se presento este error debido a que la cohorte se crea a partir de un curso.	Al revisar el código se encontró que en models en la clase de Cohorte no existía la ForeignKey del modelo Course.

6.3.2. Pruebas de Caja Blanca

Para comprobar y analizar el diseño, código, estructura interna del sistema, se realizó las pruebas de caja blanca, aplicadas y dirigidas por la desarrolladora, con el fin de encontrar algún inconveniente y mejorarlo. Las pruebas de caja blanca se las realizaron mediante test unitarios a cada uno de los módulos, en la figura 46 se puede visualizar el test realizado para iniciar sesión. Para realizar el test unitario se *(i)* especifico la ruta de conexión al método a probar, *(ii)* listar los parámetros del método e ingresar los valores que debe aceptar cada uno.

Revisar el Anexo 7, para un mejor detalle correspondiente a cada una de las pruebas de caja blanca realizadas.

```python
class TestSetUp(APITestCase):
    def setUp(self):
        self.login_url = '/auth/login/'
        self.user = User.objects.create_superuser(
            username='lola',
            email= faker.email(),
            password='lola'
        )
        response = self.client.post(
            self.login_url,
            {
                'username': 'lola',
                'password': 'lola'
            },
            format = 'json'
        )
     self.assertEqual(response.status_code, status.HTTP_200_OK)
        return super().setUp()
```

Figura 46 Test Unitario para Iniciar Sesión

```
----------------------------------------------------------------------
Ran 1 test in 1.693s

 OK
Destroying test database for alias 'default'...
```

Figura 47 Resultado del test unitario

6.3.3. Pruebas Funcionales de aceptación por el usuario

Finalizadas las pruebas de caja negra y blanca, se procedió con las pruebas de aceptación, en la cual participan los usuarios reales, encargados de verificar que el producto final cumpla con todas las tareas requeridas. Las pruebas de aceptación se llevaron a cabo en el periodo del 29 al 31 del mes de enero de 2023, conjuntamente con los usuarios finales: *(i)* Ing. Christian Salazar, *(ii)* Sra. Guadalupe Hurtado y *(iii)* dos estudiantes aleatorios, todo el proceso de estas pruebas estuvo dirigido por la desarrolladora de la codificación.

Las pruebas funcionales se detallan en tablas individuales por cada requisito, en la tabla 16 se muestra la especificación de la prueba de aceptación del registro de un pago, la cual cumplió con el objetivo de la prueba y el resultado obtenido fue exitoso, recalcando que se dio un comentario final donde se pide cambios a la interfaz.

Para un detalle más específico revisar el anexo 8 donde se contiene todas las pruebas funcionales de aceptación por parte de los usuarios.

Tabla 17 Prueba de Funcionalidad para Registrar un Pago

Requerimiento funcional:	RF 007				
Identificador: CP012	**Título**: Registrar un pago				
Detalle de la Prueba					
Fecha de realización	30-01-2023				
Requerimiento de la prueba	Registrar un pago				
Objetivo	Evaluar y verificar que el módulo de pagos permita la correcta creación de un pago.				
Descripción	El propósito de esta prueba es comprobar que el software cumple con la correcta creación de un pago.				
Hardware requerido	Computadora				
Software Requerido	Navegador Web				
Criterios de éxito	El pago se registra correctamente y se presenta un mensaje de éxito.				
Criterios de falla	Campos vacíos o datos incorrectos y presenta un mensaje exponiendo el error.				
Precondiciones	Haber iniciado sesión previamente En la lista desplegable, apartado Matrículas, buscar el estudiante al que se le registrara el pago y seleccionar la opción “Registrar Pago”				
Procedimiento de prueba	Llenar el formulario presentado. Tener en cuenta que todos los datos son obligatorios				
Parámetros para el caso de prueba	Fecha de pago Estado de pago				
Resultado esperado	Pago Registrado				
Flujo del caso de prueba	**N°**	**Usuario**	**N°**	**Sistema**	**Resultado**
			1	Busca todos los estudiantes matriculados	
			2	Carga la lista de estudiantes en la tabla “Pagos”	
	3	Selecciona un estudiante y da clic en el botón “Pagar”.			
			4	Muestra la interfaz registrar Pago	
	5	Ingresar la fecha del pago			
	6	Cambiar el estado de pago a “PAGADO”			
	7	Clic en Guardar			
			3	Valida la información ingresada	
			4	Guarda los datos	

			5	Muestra un mensaje de confirmación	
					Pago registrado con éxito
Resultado Obtenido	**Prueba Exitosa** (X)		SI ( )	NO ( )	CORRECIONES
Perfil de usuario	**Usuario**	**Nombre**		**Firma**	
	Administrador	Ing. Cristhian Salazar			
	Secretaria	Guadalupe Hurtado			
Comentarios	Resultado El usuario estuvo de acuerdo con la interfaz brindada, pero sugirió que en la tabla de pagos por cuotas se le agregue el saldo resultante después de pagar cada cuota. Solución Se solucionó colocando en la tabla de amortización un campo llamado saldo, donde se llama al valor del curso y se le va restando la cantidad mensual de cuotas hasta llegar a un saldo de 0.				

6.3.4. Pruebas de Carga y Estrés

En esta clase de pruebas, la solución informática debe estar implementado en su totalidad en un servidor real. “CENES” se encuentra alojada en servidor Centos 7 de 30 GB de espacio y 3072 MB de RAM con 200% de CPU (2V Core).

Para la ejecución de las pruebas de carga y estrés se utilizó la herramienta JMeter que permitió diseñar, ejecutar y ver en distintas formas el resultado de la ejecución del plan de pruebas. Estas pruebas fueron realizadas por la tesista a la aplicación que se encuentra ya alojada en un servidor con la siguiente dirección:

https://proeditsclub.com/cobros/frontend/index.html#/auth/login

Se describen los pasos para realizar las pruebas de carga y estrés en JMeter:

a. Se define el nombre el proyecto para la prueba de la aplicación “CENES”, como se muestra en la Figura 48.

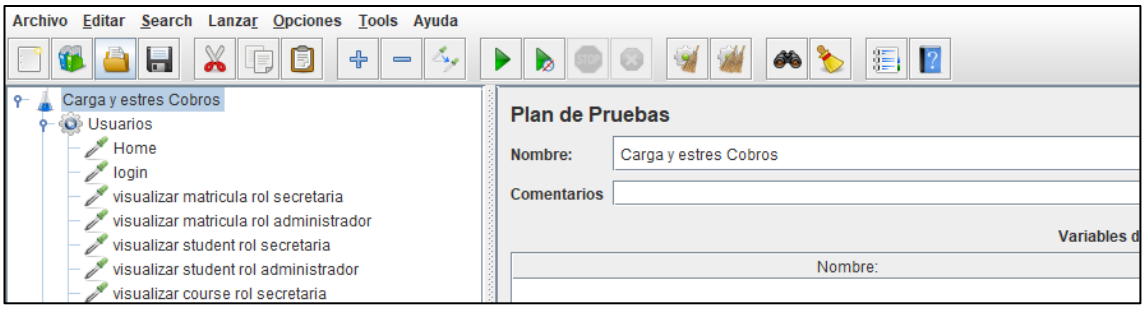


Figura 48 Creación de la prueba de Carga y Estrés

b. Crear el Grupo de Hilos donde se designó un nombre, número de hilos (número de peticiones), el periodo de subida en segundos (tiempo de ejecución) y el contador del bucle (puede ser de 1 o sin fin), como se muestra en la Figura 49.

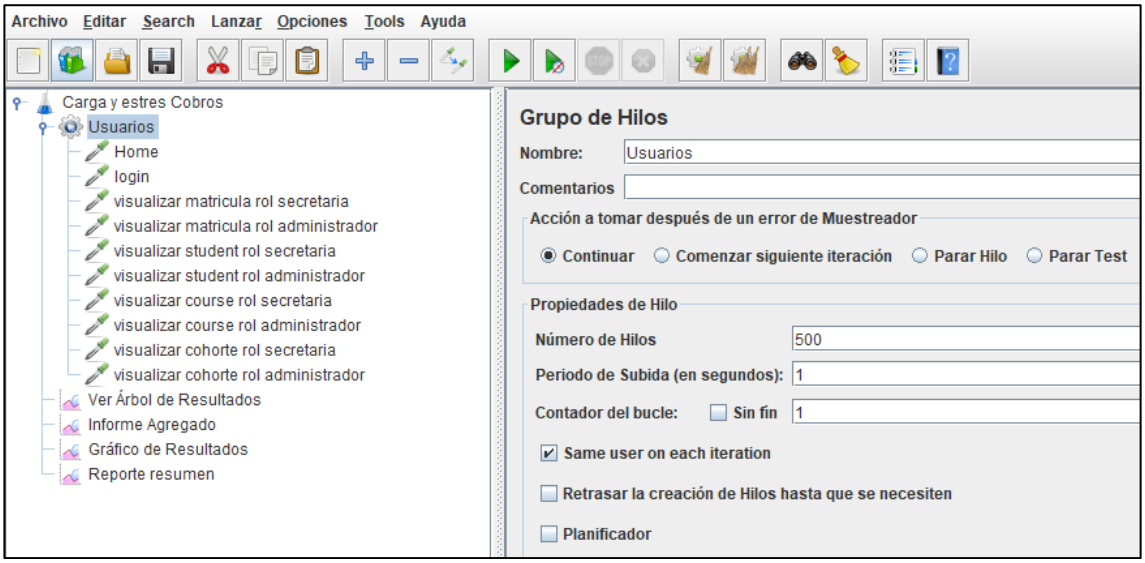


Figura 49 Creación de Grupo de Hilos para las pruebas

c. Crear la Petición HTTP, se tuvo a consideración realizar pruebas para las peticiones Get y Post que se estima que podrían producir cuellos de botella, para la configuración de las peticiones se da un nombre a la petición, se define el protocolo (Http, Https), nombre del servidor o dirección IP, tipo de petición (POST, GET), y la ruta de la petición a realizar, tal como se muestra en la figura 50.

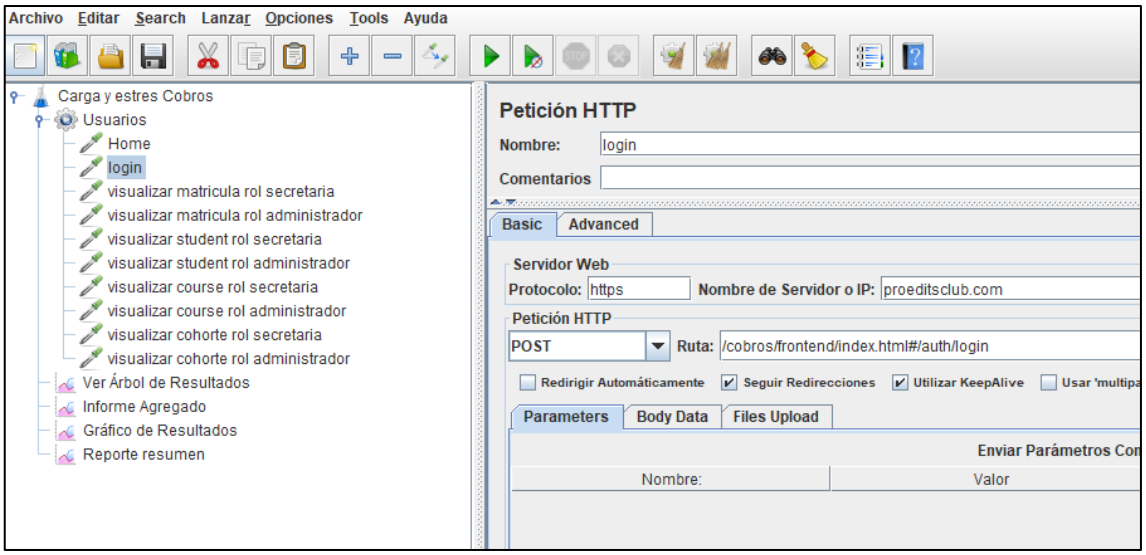


Figura 50 Creación de peticiones

En la figura 51 se muestran el gráfico de resultados donde se observa como la aplicación "CENES" al procesar las 5000 peticiones (puntos y líneas negras) por segundo logra mantener un rendimiento estable a lo largo de la ejecución de la prueba. Obteniendo un valor para la media (línea azul) de 1385, que representa el tiempo promedio en milisegundos para el conjunto de resultados, cumpliendo así el requerimiento no funcional acerca del desempeño.

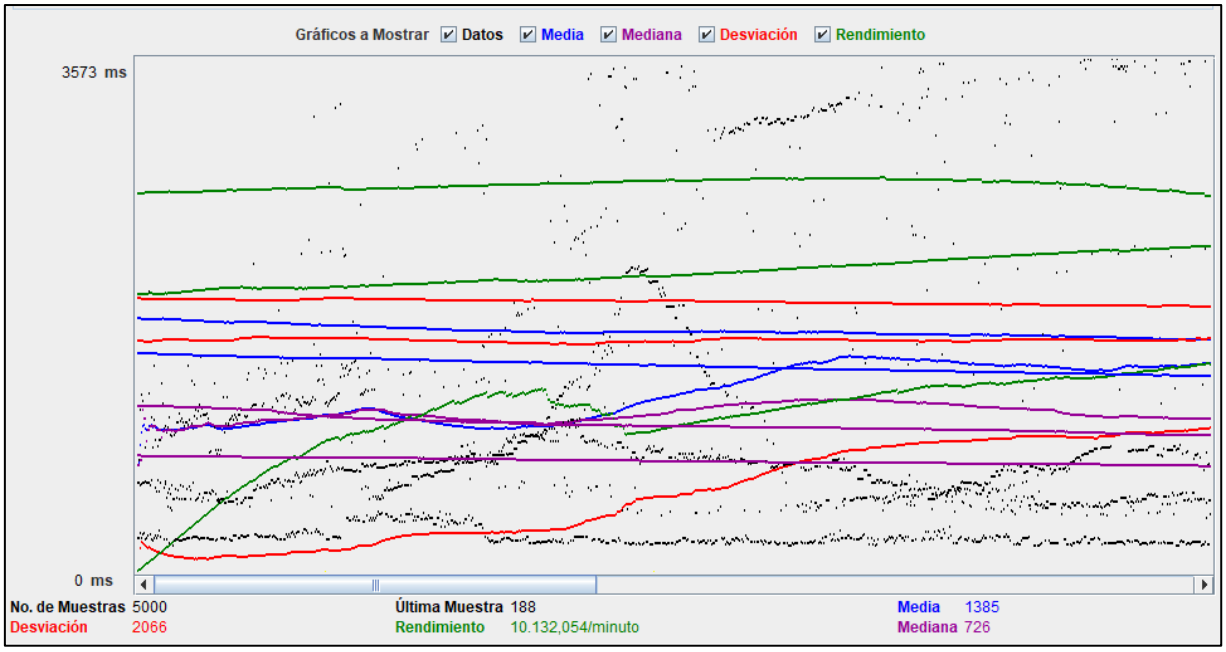


Figura 51 Gráfico de resultados de la prueba de carga y estrés

En la figura 52 se puede observar el resumen de la prueba realizada obteniendo un 0,08% de error, es decir la aplicación tiende a fallar en lo mínimo al momento de visualizar las matrículas y los estudiantes, por otro lado, se obtuvo un rendimiento de 168,9/sec, evidenciando que la aplicación puede soportar 5000 peticiones por segundo, por lo tanto, se concluye que la aplicación funciona de manera correcta en un mínimo de respuesta de 1 milisegundo y en un máximo de respuestas en 26070 milisegundos.

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Bytes
Home	500	4415	747	26070	4090,16	6,00%	18,3/sec	96,53	2,46	5411,7
login	500	1755	2	23988	2361,59	0,60%	18,5/sec	101,27	4,15	5602,8
visualizar mat...	500	1908	1	23293	1937,64	1,40%	18,7/sec	102,01	2,65	5574,5
visualizar mat...	500	1115	3	9362	1017,40	0,20%	19,3/sec	105,59	2,76	5616,9
visualizar stu...	500	1140	181	7333	995,74	0,00%	20,4/sec	112,18	2,93	5624,0
visualizar stu...	500	947	181	10832	963,25	0,00%	20,7/sec	113,54	2,97	5624,0
visualizar cou...	500	785	186	5083	722,97	0,00%	21,2/sec	116,63	3,05	5624,0
visualizar cou...	500	691	179	4255	598,19	0,00%	21,5/sec	118,27	3,09	5624,0
visualizar coh...	500	606	180	6035	656,85	0,00%	23,0/sec	126,08	3,30	5623,3
visualizar coh...	500	489	180	4373	458,02	0,00%	24,2/sec	133,17	3,48	5623,3
Total	5000	1385	1	26070	2066,49	0,82%	168,9/sec	922,64	25,42	5594,8

Figura 52 Reporte de resumen de resultados de la ejecución de la prueba carga y estrés.

En el anexo 9 se detalla el plan de pruebas de forma ampliada

7. Discusión

7.1.Desarrollo de la propuesta

El proyecto de titulación denominado “Desarrollo de aplicativo web para la gestión de cobro de mensualidades a estudiantes del preuniversitario CENES”, busca presentar una solución informática para controlar el proceso de cobros dentro de la Institución.

Objetivo 1: Documentar los requerimientos solicitados por la institución para el desarrollo del aplicativo web

Al aplicar la técnica de elicitación de la entrevista a la Sra. Guadalupe Hurtado, encargada de realizar los cobros en el preuniversitario CENES, se obtuvo como resultado el proceso de la gestión de cobros antes de aplicar la solución informática, descrito en el apartado de resultados, además, se consiguió una lista preliminar de las necesidades. Por otro lado, con el uso de la técnica de la observación se evidencio necesidades que se pudieron haber omitido en la entrevista.

Al emplear la metodología ICONIX, para la fase de análisis de requerimientos, se usó la documentación de requisitos en base al estándar IEEE-830, en el cual se determinó las necesidades del usuario, en dicho documento se expone la funcionalidad del producto mediante un caso de uso y un modelo del dominio inicial, seguidamente se describe los usuarios que participaran en la solución informática, cada uno con su rol especificado y una descripción de los permisos permitidos, además, se define en profundidad 10 requisitos funcionales y 4 no funcionales, se estableció un diccionario de datos con términos claves que permitan entender el concepto y contexto de la documentación así como de la solución informática, el anexo 2 cuenta con la especificación completa de los requisitos.

La especificación de requisitos se puntualizó bajo el estándar IEEE-830, debido a que, en comparación con el estándar IEEE 29148, este proporciona un esquema puntual que guía para especificar cada requisito de forma general, permitiendo en primer lugar que dichos requisitos sean aprobados por el cliente, para posteriormente detallar el funcionamiento de cada uno de ellos.

Objetivo 2: Diseñar el modelo arquitectónico en base a los requerimientos con el fin de obtener la interfaz gráfica del aplicativo web

Para el diseño del modelo arquitectónico, en primer lugar se realizó la fase de análisis y diseño preliminar de la solución informática, se realizó un detalle de cada caso de uso, dicho detalle contiene: *(i)* prototipo de pantalla y *(ii)* la descripción del flujo a cumplir, posteriormente se procedió al diseño del diagrama de robustez con el fin de comprobar un comportamiento razonable de los casos de uso, asegurándose que en los casos de uso se cubra el proceso básico y caminos alternos, además, se actualizo el diagrama de dominio en el cual ya se especifican los principales datos encontrados.

A partir del análisis y el prototipado se continuo con la fase de diseño, modelando el estilo arquitectónico de la solución en base a las características de funcionalidad y de no funcionalidad, se determinó un tipo de arquitectura Cliente-Servidor y REST, procediendo con el diagrama de secuencia, en este se detalla el procedimiento que debe cumplir cada caso para generarse con éxito, de igual manera, se detalló el diagrama de clases en el cual se determinan los datos que contendrá cada clase, la relación entre ellas y la cardinalidad, procediendo con el diagrama de componentes donde se obtuvo 4 componentes: web, móvil, servicio web y base de datos, y finalmente, para un vista física se realizó el diagrama de despliegue, donde se puede observar un equipo web y un móvil, así como un servidor web y uno de base de datos, todo lo anterior mencionado se encuentra expuesto más a profundidad en el apartado de resultados.

Se continuo con la fase de implementación, en primer lugar, se realizó la codificación de la solución, con un patrón de diseño Modelo Vista Controlador, para el BackEnd se usó el lenguaje de programación Python conjuntamente con el framework Django Rest para el desarrollo de la lógica del aplicativo y la creación de las API´s que permiten realizar la correcta comunicación con el FrontEnd.

El FrontEnd se divide en dos: *(i)* FrontEnd Web y *(ii)* FrontEnd móvil, *(i)* se codificó en Angular el mismo que se basa en HTML, CSS y TypeScript, al basarse en JavaScript, utiliza NodeJS, para la ejecución del servidor, *(ii)* se desarrolló en el framework de Flutter que se basa en el lenguaje de programación Dart, cabe recalcar que la aplicación móvil se ha generado específicamente para Android dado que no se dispone del hardware necesario para realizar pruebas en dispositivos IOS.

Cabe recalcar que se usó un diseño arquitectónico REST, conjuntamente con la arquitectura cliente-servidor ya que van de la mano, con el fin de separar responsabilidades entre el cliente y el servidor, trabajando así con tecnologías

heterogenias como lo son: para el lado el cliente web el framework Angular, flutter para el cliente móvil y para el lado del servidor se utilizó django restframework, el que se conecta a una base de datos PostgreSQL. Al existir una separación entre el cliente y el servidor, se da la posibilidad de que el producto final sea escalable, el desarrollo de las diferentes partes del proyecto se puede dar de manera independiente, lo cual es una ventaja a futuro, cuando la empresa empiece a crecer.

De esta manera se obtuvo una arquitectura detallada y precisa, lo que permitió mostrar físicamente como debe comportarse la solución informática desarrollada.

Objetivo 3: Evaluar el aplicativo web en un ambiente simulado

Al terminar con la codificación, se prosiguió a evaluar el aplicativo web, para cual se planificó una serie de pruebas: *(i)* Prueba de caja negra, *(ii)* Prueba de Caja Blanca, *(iii)* Prueba de Funcionalidad y *(iv)* Prueba de carga y estrés. Iniciando con las pruebas de caja negra, comprobando que los datos de entrada hacia la lógica del aplicativo devuelven datos de salida de manera correcta, cabe recalcar que se encontraron algunos errores, los mismos que fueron solucionados posteriormente. Se prosiguió a realizar pruebas de caja blanca con el fin de identificar posibles errores de codificación, utilizándolos test unitarios para cada uno de los métodos, Python brinda un paquete de test que permite ejecutar las pruebas sin mucho esfuerzo. Continuando con las pruebas de funcionalidad aplicadas a los usuarios finales, quienes experimentaron el manejo del aplicativo web ya funcional, en estas pruebas, se pudo determinar pequeños controles que no se estaba tomando en cuenta, los que fueron solucionados con éxito. Finalmente, se realizaron las pruebas de carga y estrés, el aplicativo fue sometido a 500 peticiones en tiempo real, el resultado de las pruebas de que el rendimiento del sistema como su disponibilidad y estabilidad bajo esta carga se mantuvo estable.

Se determino aplicar pruebas de caja negra y blanca con el fin de ayudar a encontrar errores que no dejen tener un correcto funcionamiento del software, en las pruebas se encontraron algunos errores de código, que al momento de realizar la codificación se pasaron por alto, pero gracias al uso de estas pruebas se pudo corregir a tiempo. Jmeter es un software con bastante popularidad y acogida al momento de realizar pruebas de carga y estrés, es por ello que se lo utilizó, con el fin de determinar la cantidad de usuario concurrentes que puede manejar el aplicativo desarrollado y de mostrar su rendimiento mediante un análisis gráfico, lo que permitió tener una idea más clara en cuanto a resultados.

7.2.Valoración Técnica Económica Ambiental

En base a recursos técnicos, económicos y ambientales se realizó el presente Trabajo de Titulación.

7.3.Valoración Técnica

Para el desarrollo de TT se usó la valoración técnica de varias tecnologías que trabajando en conjunto se logró el cumplimiento de los objetivos:

- Herramientas Gestor Bibliográfico Mendeley para las citas contempladas dentro del TT.
- Servicios como lo son: Google Drive y One Drive para el manejo de documentación, Draw.io/Diagramas.net y Enterprise Architect para diseñar diagramas, Balsamiq para diseñar los prototipos, Mensajería instantánea para la constante comunicación con los Stakholders y el director de tesis.
- Frameworks como los son: Django Rest, Angular y Flutter para el desarrollo de la solución informática.

7.4.Valoración Ambiental

Al usar recursos digitales para el desarrollo del TT, se redujo en gran parte el uso de medios de transporte y materiales de oficina aportando al cuidado del entorno en el que habitamos.

7.5.Valoración Económica

Es importante mencionar que, para el desarrollo de presente TT, así como del aplicativo web, se utilizó software completamente gratuito, el financiamiento del director de tesis y tutor académico son cubiertos por la Universidad Nacional de Loja, los demás gastos fueron cubiertos por la autora del presente TT.

8. Conclusiones

Finalizado el TT se concluye lo siguiente:

- El desarrollo del aplicativo web para la gestión de cobros permitió reducir las inconsistencias en la información y la tasa de incumplimiento al mostrar un reporte de los alumnos cuentan con cuotas vencidas, para proceder a realizar el cobro y registro del pago respectivo.

- La aplicación de la técnica de elicitación de entrevista fue punto clave para determinar, recopilar y detallar 10 requisitos funcionales y 4 requisitos no funcionales que la solución informática debe contemplar, lo cuales quedaron establecidos en el documento de Especificación de Requisitos en base al estándar IEEE 830.

- El análisis de los requerimientos conjuntamente con los artefactos de diseño fue una gran ayuda para la organización del modelo arquitectónico, el que se basa en cliente-servidor y REST, todo esto con el fin de tener una mejor comprensión y comunicación al momento de desarrollar la solución informática.

- La metodología ICONIX es apropiada para proyectos de mediano alcance en procesos de cobros, debido a que el tiempo de desarrollo de estos es corto, se adapta fácilmente a trabajar conjuntamente con el cliente, lo cual ayuda a eliminar trabajo innecesario. Además, presenta de manera clara cuáles son las actividades a desarrollar en cada etapa, lo que permite tener la capacidad de seguir una relación entre los diferentes artefactos producidos en cada fase hasta llegar al desarrollo exitoso del software.

- En la fase de pruebas, las pruebas de caja negra y blanca ayudaron a identificar el mayor porcentaje de errores de funcionamiento en el módulo de gestión de cobros permitiendo dar las respectivas correcciones y de esta manera se reduzcan los problemas de funcionalidad.

- El modelo arquitectónico de Rest permitió integrar las tecnologías del heterogéneas desde el backend, al frontend tales como: Django RestFramework, Angular y Flutter, dando como resultado la unión de estas tecnologías el producto final que es el aplicativo web.

9. Recomendaciones

Finalizado el TT se recomienda lo siguiente

- En el desarrollo considerar siempre las seguridades para lo que es el caso de Django resframework utilizar tokens para controlar el inicio de sesión y conjuntamente con angular trabajar lo que es el encapsulamiento de datos.

- En este tipo de proyecto se utilicé una metodología de validación basada en V, desde la primera fase de especificaciones hasta el montaje de la aplicación, para minimizar riesgos y mejorar la planificación a través de los roles y resultados predeterminados.

- Se extienda un módulo de cobros que permita registrar pagos y este se base en paypel para que la consistencia de la información sea más clara, precisa y confiable.

- Agregar la funcionalidad de notificación por correo/celular con un recordatorio de que su mensualidad esta próxima a vencer.

10. Bibliografía

[1] C. A. Vergara Rosas, “Sistema Web para la gestión de cobranza en la empresa Soseka SAC.”

[2] J. V. SALINAS COBEÑA, “la gestión de cobranzas frente a la emergencia sanitaria de la empresa Casaplan Motorplan s.a. del cantón la libertad, provincia de Santa Elena.”

[3] V. Y. Hans Richard, “Arquitectura de software basada en microservicios para la implementación de la aplicación web de cobranza digital den Financial Systems Company SAC,” 2019.

[4] E. Velasteguí and C. Sánchez, “La gestión de cobranza y la automatización por una base de datos.,” *Visionario Digital*, vol. 1, no. 3, pp. 41–54, Jul. 2017, doi: 10.33262/visionariodigital.v1i3.257.

[5] J. Mark, “El código de Hammurabi,” *https://www.worldhistory.org/trans/es/1-19882/el-codigo-de-hammurabi/*, Jun. 24, 2021.

[6] C. Ayala, “Los fenicios, inventores del comercio internacional,” *https://bienpensado.com/breve-historia-ventas-fenicios-comercio-internacional/*, Aug. 30, 2018.

[7] J. Jiménez Bolaños, “la obligación civil romana y las garantías del derecho de crédito,” *Revista Judicial*, 2013.

[8] L. A. Arias López, D. G. Flores González, and I. L. Rivas Callejas, “La falta de ley especial reguladora de las tarjetas de débito y crédito en defensa del consumidor en el cobro no pactado por servicios en el contrato de emisión por los bancos,” 2013.

[9] RSM, “Beneficios de la tecnología en la gestión de cobranzas,” *https://www.rsm.global/peru/es/aportes/blog-rsm-peru/como-se-aplica-la-tecnologia-en-la-gestion-de-cobranzas#*, Feb. 15, 2019.

[10] J. M. Osuna Quintero, “Proceso de mejora en procedimiento de crédito y cobranza,” 2018.

[11] E. E. del Valle Córdova, “crédito y cobranzas.”

[12] L. Díaz and D. Y. Flores Enríquez, “Gestión de Cobranza: un abordaje teórico desde el ámbito financiero,” *Maya - Revista de Administración y Turismo*, vol. 1, no. 1, pp. 56–68, Apr. 2021, doi: 10.33996/maya.v1i1.6.

[13] “Descuento Simple.”

[14] J. G. Pascual and R. V. Gomes Bastos, “La morosidad; un acuciante problema financiero de nuestros días.”

[15] G. Lincoyán Portus, “Matemáticas Financieras,” 1997.

[16] F. J. Garcia Peñalvo and A. Garcia Holgado, “Fundamentos De La Vista De Casos De Uso,” *Departamento de Informática y Automática Universidad de Salamanca*, pp. 2–36, 2017.

[17] I. Jacobson, “Object oriented development in an industrial environment,” *Conference Proceedings on Object Oriented Programming Systems Languages and Applications, OOPSLA 1987*, vol. 1987-Janua, pp. 183–191, 1987, doi: 10.1145/38765.38824.

[18] R. M. Agut, “Especificación de Requisitos Software según el estándar de IEEE 830,” 2000.

[19] S. Engineering and S. Committee, *IEEE recommended practice for software requirements specifications*, vol. 1998, no. October. 2011. doi: 10.1109/9781118156674.ch3.

[20] C. R. P. de San Martin Olivia, “Uso de la Metodología ICONIX.”

[21] kARENNY BRITO ACUÑA, “Selección de metodologías de desarrollo para aplicaciones web en la facultad de informática de la Universidad de Cienfuegos,” Universidad Cienfuegos, Cuba, 2009.

[22] python, “El Libro de Python,” *https://ellibrodepython.com/que-es-python*. https://ellibrodepython.com/que-es-python (accessed Oct. 30, 2022).

[23] A. Robledano, “Qué es Python: Características, evolución y futuro,” *https://openwebinars.net/blog/que-es-python/*, Sep. 2019. https://openwebinars.net/blog/que-es-python/ (accessed Oct. 30, 2022).

[24] Django Software Foundation, “Meet Django,” *https://www.djangoproject.com/*.

[25] A. Holovaty and J. Kaplan, “El Libro de Django,” 2008.

[26] MkDocs, “Marco REST de Django.”

[27] C. X. Peraza de Aparicio and N. Y. Zurita Barrios, “Las bases de datos como estrategia didáctica para investigadores noveles,” *RECIMUNDO*, vol. 4, no. 4, pp. 19–29, Nov. 2020, doi: 10.26820/recimundo/4.(4).noviembre.2020.19-29.

[28] PostgreSQL Global Development Group, “ What Is PostgreSQL?,” *https://www.postgresql.org/docs/current/intro-whatis.html*, Oct. 13, 2022.

[29] P. López, H. Patricia, and A. Uk, “View metadata, citation and similar papers at core.”

[30] M. Boada Oriols and J. A. Gómez Gutiérrez, “El gran libro de Angular,” 2019.

[31] D. Villalón Pardo, “Crear y desarrollar una aplicación de alto rendimiento con bajo coste utilizando Flutter y Firebase,” 2021.

[32] V. Vázquez Rodríguez, “Desarrollo de aplicaciones móviles multiplataforma con Flutter,” 2019.

[33] Y. P. FLOREZ TABORDA, “Desarrollo de una aplicación móvil para la búsqueda de sitios turísticos utilizando la georreferenciación del municipio de Quinchía Risaralda,” 2021.

[34] M. Maldonado del Toro, “Repositorio de Componentes para el Departamento de Servicios Informáticos para Internet,” 2015.

[35] B. Kaseng Solís, F. Lizardo, M. Rubén, and E. H. Ochante, “‘Sistema web para el proceso de cobranza en la empresa de créditos Sebastián,’” 2019.

[36] E. D. Amaya Lozano and C. S. Juez Cadell, “Análisis, diseño, desarrollo e implementación de un sistema de control para registros y cobro de matrícula y pensiones para la unidad educativa particular mixta Mercedes de Jesús Molina mediante un aplicativo web.,” UNIVERSIDAD POLITÉCNICA SALESIANA, GUAYAQUIL, 2016.

[37] L. E. Marin Puris, “Sistema Web para el control de pagos en la I.E.P. Diego Thomson de Mangomarca, S.J.L 2017,” Universidad Cesar Vallejo, Lima, 2017.

[38] L. M. Alejandro Camacho, “Desarrollo e implementación de una aplicación web para la administración de pagos de agua del barrio el zarza-cantón Yantzaza,” Zamora Chinchipe, Aug. 2021.

11. Anexos

Anexo 1. Entrevista

Acta N° 001

CIUDAD	LOJA	**FECHA**	09 de noviembre de 2022	**HORA**	17:30 PM

OBJETO DE LA REUNIÓN:	
Reunión para entrevistar a la persona encargada del proceso de cobros en el preuniversitario CENES.	
ASISTENTES	
NOMBRE	**CARGO**
Guadalupe Hurtado	Secretaria del preuniversitario CENES, encargada del cobro de mensualidades.
Carla Isabel Troya Capa	Tesista, Estudiante de la carrera de Ingeniería en Sistemas/Computación.
AUSENTES	

ORDEN DEL DÍA

Una vez verificada la asistencia del quórum reglamentario para deliberar y decidir, se sometió a consideración el orden del día propuesto:

1. Explicación del proceso a llevar a cabo para la entrevista
2. Entrevista
3. Varios.

DESARROLLO

1. **Explicación del proceso a llevar a cabo para la entrevista**:
   Se explica la dinámica para la entrevista a la persona encargada del proceso de cobros en el preuniversitario CENES la Sra. Guadalupe Hurtado, la estudiante realizará la pregunta y ella responderá lo más concretamente posible.

2. **Entrevista**
   2.1. **¿Cuál es el personal involucrado y que rol cumple cada uno en el proceso de difusión de eventos?**
   Se parte de una planificación acorde al calendario académico.

   Dos involucrados principales:

Gerente General: Planifica, realiza el Seguimiento, Control y Cierre de los cursos a ofertados.

Encargado del cobro: una vez planificados los cursos del calendario académicos se da autorización a la secretaría para el cobro a los estudiantes que se inscriban a dichos cursos.

Además, se tiene otros involucrados como lo son:

Cliente (Estudiante/Representante que se encarga del pago de mensualidades)

2.2. ¿Cómo se está llevando actualmente el proceso del cobro de mensualidades en el preuniversitario?

El proceso se base en las siguientes fases:

- Llenar manualmente una ficha con los datos del estudiante, especificando el curso que va a seguir.
- Preguntar al estudiante si el cobro será al contado o en cuotas.
- En caso de ser cuotas la secretaría se encarga de realizar el cálculo dependiendo de los meses que quiera la diferir el pago, según el curso que seguirá.
- Para la inscripción el estudiante debe dejar pagando 20 dólares.
- Se entrega un comprobante del cobro, a partir de ahí puede asistir con normalidad al curso.

Adicionalmente se tiene en cuenta que:

- Existen descuentos para abanderados y escoltas.
- Para realizar los cobros que se establecieron en cuotas se debe estar en una constante revisión de las fichas estudiantiles para saber el día exacto en el que debe realizarse el cobro.

2.3. ¿Existe procesos o procedimientos que se hayan creado para la ayuda de cobros de mensualidades?

No se cuenta con ningún proceso, más que el mencionado anteriormente.

2.4. ¿Cuáles considera usted que son las diferencias entre la forma en que opera actualmente el proceso y la forma en que debería operar?

Actualmente no opera al 100%, es muy inconsistente debido al proceso que

se realiza, ya que para poder llevar un seguimiento a los deudores se tiene que recurrir a la revisión de múltiples documentos, lo que implica que se emplee demasiado tiempo.

Por lo cual la forma que debería operar el proceso sería que nos permita ingresar los datos de los estudiantes, tener presente sus cuotas y fechas a pagar, así como un reporte final, reduciendo de esta manera tiempos de búsqueda y retrasos en los pagos por parte de los clientes.

2.7. ¿Cuáles considera usted que son los principales problemas en el proceso actual?

Existen varios problemas en el actual proceso, desde la pérdida o duplicidad de fichas, lo que se convierte en un problema a la hora de revisar los cobros correspondientes, lo cual involucra que se debe revisar uno por uno los recibos que se han entregado a partir del inicio del curso.

Y para tener un reporte final, toda esa información se debe ingresar en un archivo de Excel, lo que requiere de tiempo muy considerable.

2. Varios

Tener en cuenta que existen estudiantes que ingresan 1 o 2 meses después, por lo cual, si cancela en cuotas las cuotas se reducen en cuanto a los meses restantes que quedan del curso.

Existen diferentes cursos, cada uno con diferente duración en cuanto a meses.

En constancia firman:

NOMBRE	CARGO	FIRMA
Guadalupe Hurtado	Secretaria del preuniversitario CENES, encargada del cobro de mensualidades.	
Carla Isabel Troya Capa	Tesista, Estudiante de la carrera de Ingeniería en Sistemas/Computación.	

Anexo 2. Especificación de Requisitos

Desarrollo de aplicativo web para la gestión de cobro de mensualidades a estudiantes del preuniversitario CENES

Especificación de requisitos de software

FICHA DEL DOCUMENTO

HOJA DE CONTROL

Organismo	Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables
Proyecto	Desarrollo de aplicativo web para la gestión de cobro de mensualidades a estudiantes del preuniversitario CENES
Entregable	Especificación de Requisitos
Autor	Carla Troya

Documento validado por las partes en fecha:

Por la comunidad	Por la universidad
Carrera de Ingeniería en Sistemas/Computación.	Universidad Nacional de Loja

Tabla de Contenido:

Índice de Tablas

1. Introducción

Este documento es una Especificación de Requisitos Software (ERS) del aplicativo web para la gestión de cobro de mensualidades a estudiantes del preuniversitario CENES. Esta especificación se ha estructurado basándose en las directrices dadas por el estándar IEEE ANSI/IEEE 830, 1998[1].

1.1.Propósito

El presente documento tiene como propósito definir las especificaciones funcionales, no funcionales para el desarrollo del aplicativo web que permitirá gestionar el proceso para realizar los cobros mensuales a los estudiantes del preuniversitario CENES.

1.2.Alcance

Esta especificación de requisitos está dirigida al usuario del aplicativo web, para profundizar en la automatización de la solución, la cual tiene por objetivo principal el gestionar el proceso para el cobro de mensualidades a los estudiantes de la institución.

1.2.1. Personal involucrado

Tabla 1 Personal Involucrado

Nombre	Carla Isabel Troya Capa
Rol	Analista, diseñador y programador
Categoría Profesional	Estudiante de ingeniería en Sistemas
Responsabilidad	Análisis de información, diseño y programación del aplicativo.
Información de contacto	citroyac@unl.edu.ec

1.2.2. Definiciones, acrónimos y abreviaturas

Tabla 2 definiciones, acrónimos y abreviaturas

Nombre	Descripción
Aplicativo web	Es un conjunto de páginas dinámicas cuyo contenido se determina después que un usuario haya interactuado con ella accediendo a un servidor web a través de Internet o de una intranet mediante un navegador.
Usuario	Persona que usará el sistema para gestionar el proceso
ERS	Especificación de Requisitos Software
RF	Requerimiento Funcional
RNF	Requerimiento No Funcional
CENES	Centro de Nivelación Estudiantil
COHORTE	Son las opciones que tendrá un curso (Mañana, Tarde, etc.)

1.2.3. Referencias

Tabla 3 Referencias

Título del Documento	Referencia
Standard IEEE 830 - 1998	IEEE [1] S. Engineering and S. Committee, *IEEE recommended practice for software requirements specifications*, vol. 1998, no. October. 2011.

2. Resumen

Este documento consta de tres secciones. En la primera sección se realiza una introducción al mismo y se proporciona una visión general de la especificación de recursos del aplicativo web.

En la segunda sección del documento se realiza una descripción general del aplicativo web, con el fin de conocer las principales funciones que éste debe realizar, los datos asociados y los factores, restricciones y dependencias que afectan al desarrollo, sin entrar en excesivos detalles.

Por último, la tercera sección del documento es aquella en la que se definen detalladamente los requisitos que debe satisfacer el aplicativo web.

3. Descripción general

3.1. Perspectiva del producto

El sistema web para la gestión de cobros será un producto diseñado para el registro de pagos, lo que permitirá su utilización de forma rápida y eficaz.

A continuación, se puede observar en el caso de uso que se presenta en el siguiente punto.

3.1.1. Funcionalidad del Producto

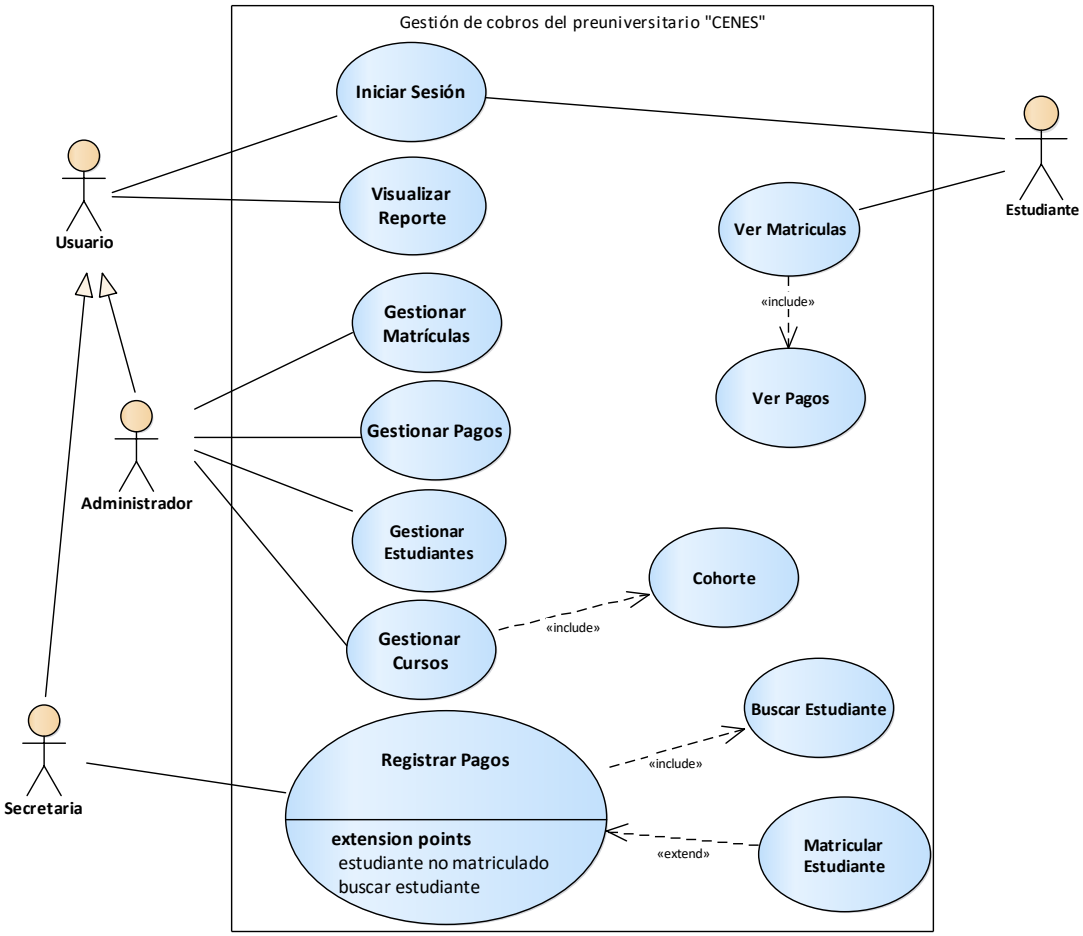


Figura 1 Caso de uso para la gestión de cobros

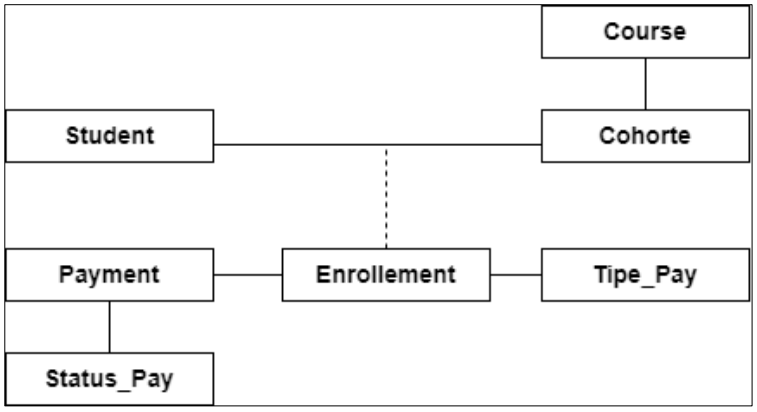


Figura 2 Modelo del dominio inicial

3.2. Características de los usuarios

Tabla 4 Tipo de Usuario: "Administrador"

Tipo de usuario	Administrador
Formación	Educador
Actividades	Control y manejo del sistema según su usuario asignado

Tabla 5 Tipo de Usuario: "secretaria"

Tipo de usuario	Secretaria
Formación	Contadora
Actividades	Control y manejo del sistema según su usuario asignado

Tabla 6 Tipo de Usuario: "Estudiante"

Tipo de usuario	Estudiante
Formación	Estudiante
Actividades	Manejo del sistema según su usuario asignado

3.2.1. Restricciones

- El sistema deberá tener un diseño e implementación sencilla, independiente de la plataforma o del lenguaje de programación.

4. Requisitos específicos

4.1. Requerimientos funcionales

Tabla 7 Requerimiento Funcional 1

Identificación del requerimiento:	RF001
Nombre del Requerimiento:	Iniciar sesión
Descripción del requerimiento:	El aplicativo permitirá al usuario (administrador, secretaria, estudiante) iniciar sesión y dependiendo del tipo de usuario le presentarán las funcionalidades.
Requerimiento NO funcional:	• RNF01, RNF02, RNF03, RNF04, RNF05
Prioridad del requerimiento: Alta	
Dependencia del requerimiento:	

Tabla 8 Requerimiento Funcional 2

Identificación del requerimiento:	RF002
Nombre del Requerimiento:	Gestionar curso
Descripción del requerimiento:	El aplicativo permitirá al usuario (Administrador) crear, editar, buscar y eliminar los cursos que ofrece la institución. Cada Curso tiene una o varias Cohortes creadas. Para eliminar un curso se debe comprobar que no existan cohortes creadas a partir de este.
Requerimiento NO funcional:	• RNF01, RNF02, RNF03, RNF04, RNF05
Prioridad del requerimiento: Alta	
Dependencia del requerimiento: RF001	

Tabla 9 Requerimiento Funcional 3

Identificación del requerimiento:	RF003
Nombre del Requerimiento:	Gestionar cohorte

Descripción del requerimiento:	El aplicativo permitirá al usuario crear, editar, buscar y eliminar una cohorte. Tener en cuenta que la cohorte se crea a partir de un curso y para poder eliminar una cohorte se debe comprobar que no existan estudiantes matriculados en ella.
Requerimiento NO funcional:	• RNF01, RNF02, RNF03, RNF04, RNF05
Prioridad del requerimiento: Alta	
Dependencia del requerimiento: RF001, RF003	

Tabla 10 Requerimiento Funcional 4

Identificación del requerimiento:	RF004
Nombre del Requerimiento:	Crear un estudiante
Descripción del requerimiento:	El aplicativo web permitirá a los usuarios (Administrador y secretaria) llenar un formulario, el mismo que pedirá los datos personales y necesarios para su inscripción.
Requerimiento NO funcional:	• RNF01, RNF02, RNF03, RNF05
Prioridad del requerimiento: Alta	
Dependencia del requerimiento: RF01	

Tabla 11 Requerimiento Funcional 5

Identificación del requerimiento:	RF005
Nombre del Requerimiento:	Editar datos del estudiante
Descripción del requerimiento:	El aplicativo web permitirá al usuario (Administrador) visualizar y editar los datos del estudiante de acuerdo a las necesidades presentadas.
Requerimiento NO funcional:	• RNF01, RNF02, RNF03, RNF04, RNF05
Prioridad del requerimiento: Alta	
Dependencia del requerimiento: RF001, RF003	

Tabla 12 Requerimiento Funcional 6

Identificación del requerimiento:	RF006
Nombre del Requerimiento:	Matricular estudiantes
Descripción del requerimiento:	El aplicativo permitirá al usuario (administrador y secretaria) matricular al estudiante en un curso. Además, admitirá eliminar una matrícula, al eliminar dicha matrícula se eliminará también todos los pagos en el caso de que se hayan realizado.
Requerimiento NO funcional:	• RNF01, RNF02, RNF03, RNF04, RNF05
Prioridad del requerimiento: Alta	
Dependencia del requerimiento: RF001, RF002, RF003	

Tabla 13 Requerimiento Funcional 7

Identificación del requerimiento:	RF007
Nombre del Requerimiento:	Registrar pago
Descripción del requerimiento:	El aplicativo permitirá a los usuarios (secretaria y administrador) registrar el pago de acuerdo a cada uno de sus estudiantes matriculados, el pago puede ser en cuotas o al contado. Si el pago es en cuotas el sistema deberá calcular el monto a pagar dependiendo el número de cuotas. En caso de que el estudiante no conste en el sistema, se lo debe registrar y matricular (RF002).
Requerimiento NO funcional:	• RNF01, RNF02, RNF03, RNF05
Prioridad del requerimiento: Alta	
Dependencia del requerimiento: RF001, RF006	

Tabla 14 Requerimiento Funcional 8

Identificación del	RF008

requerimiento:	
Nombre del Requerimiento:	Editar pagos
Descripción del requerimiento:	El aplicativo web permitirá a los usuarios editar y visualizar cualquier pago registrado. Con el fin de que si ocurre una confusión de pagos puedan ser corregidas.
Requerimiento NO funcional:	• RNF01, RNF02, RNF03, RNF04, RNF05
Prioridad del requerimiento: Media	
Dependencia del requerimiento: RF001, RF007	

Tabla 15 Requerimiento Funcional 9

Identificación del requerimiento:	RF009
Nombre del Requerimiento:	Visualizar Reporte
Descripción del requerimiento:	El aplicativo web permitirá a los usuarios visualizar un reporte con un listado de los estudiantes y sus pagos.
Requerimiento NO funcional:	• RNF01, RNF02, RNF03, RNF04, RNF05
Prioridad del requerimiento: Media	
Dependencia del requerimiento: RF001, RF003, RF006	

Tabla 16 Requerimiento Funcional10

Identificación del requerimiento:	RF010
Nombre del Requerimiento:	Visualizar matrículas
Descripción del requerimiento:	El aplicativo permitirá al usuario estudiante visualizar los cursos en los que se encuentra matriculado y sus pagos realizados por cada curso.
Requerimiento NO funcional:	• RNF01, RNF02, RNF03, RNF04, RNF05
Prioridad del requerimiento: Media	
Dependencia del requerimiento: RF001, RF003, RF006	

4.2. Requerimientos no funcionales

Tabla 17 Requerimiento No Funcional 1

Identificación del requerimiento:	RNF01
Nombre del Requerimiento:	Interfaz del sistema.
Descripción del requerimiento:	El aplicativo web debe tener una interfaz lo más intuitiva y sencilla que permita la herramienta, para que sea de fácil manejo a los usuarios del módulo.
Prioridad del requerimiento:	Alta

Tabla 18 Requerimiento No Funcional 2

Identificación del requerimiento:	RNF02
Nombre del Requerimiento:	Ayuda en el uso del sistema.
Descripción del requerimiento:	La interfaz del aplicativo debe ser amigable para el usuario y de fácil comprensión, con la finalidad que pueda acceder a las diferentes funcionalidades de forma rápida e intuitiva.
Prioridad del requerimiento:	Alta

Tabla 19 Requerimiento No Funcional 3

Identificación del requerimiento:	RNF03
Nombre del Requerimiento:	Diseño de la interfaz vinculada con el de la institución.
Descripción del requerimiento:	La interfaz de usuario debe ajustarse en lo más posible permitido a los colores de la institución.
Prioridad del requerimiento:	Alta

Tabla 20 Requerimiento No Funcional 4

Identificación del requerimiento:	RNF04
Nombre del Requerimiento:	Desempeño
Descripción del requerimiento:	Garantizar el desempeño del aplicativo web a los diferentes usuarios. En este sentido la información podrá ser consultada y actualizada permanente y simultáneamente.
Prioridad del requerimiento: Alta	

Tabla 21 Requerimiento No Funcional 5

Identificación del requerimiento:	RNF05
Nombre del Requerimiento:	Seguridad en información
Descripción del requerimiento:	Garantizar la seguridad del aplicativo web con respecto a la información y datos que se manejan tales sean documentos, archivos.
Prioridad del requerimiento: Alta	

5. Requisitos comunes de las interfaces

5.1.1. Requisitos de rendimiento

- Se asume que los requisitos aquí descritos son estables
- Los equipos en los que se vaya a ejecutar el aplicativo deben cumplir los requisitos para garantizar una ejecución correcta de la misma.
    - Requisitos de Hardware:
        - Mínimo Procesador core i5
        - Mínimo 4 GB de memoria RAM
        - Mínimo 10 GB de espacio de almacenamiento
    - Requisitos de Software:
        - Sistema Operativo:
            - Microsoft Windows Server 64 bits, 2016
            - Ubuntu 18.04 LTS 64 bits
        - Base de Datos:
            - PostgreSQL 11.2 o superior
            - Servidor SQL 2017
        - Navegador:
            - Google Chrome
            - Mozila Firefox
            - Michosft Edge

5.1.2. Interfaces de usuario

La interfaz de usuario constará de un conjunto de ventanas con botones, listas y campos de textos.

5.1.3. Interfaces de hardware

Será necesario disponer de equipos de cómputos en perfecto estado con las siguientes características:

- Adaptadores de red.
- Procesador de 4 núcleos o superior.
- Memoria mínima de 4GB
- Mouse.

- Teclado.

5.1.4. Interfaces de software

- Sistema Operativo: Adaptativo.
- Explorador: Adaptativo.

En constancia firman:	
Ing. Christian Mauricio Salazar Valle GERENTE GENERAL DE "CENES" **(Cliente)**	Carla Isabel Troya Capa **Responsables de la especificación de requisitos**

CENES
CENTRO DE NIVELACIÓN ESTUDIANTIL
R.U.C. 1719746842001
LOJA - ECUADOR

Especificación de Casos de Uso

Desarrollo de aplicativo web para la gestión de cobro de mensualidades a estudiantes del preuniversitario "CENES"

Fecha: *14/11/2022*

Tabla de Contenido

1. Resumen

En el presente documento se puede observar tres secciones. En la primera sección se puede observar el caso de uso general diseñado para el proceso de gestión de cobros.
En la segunda sección, se detalla cado uno de sus actores y la relación que tienen entre ellos. Por último, la tercera sección de documento es la especificación de cada uno de los casos de uso que forman parte del proceso para la gestión de cobros, detallando el flujo que se debe seguir para que se realicen con éxito las tareas necesarias para el proceso.

2. Diagrama de Casos de Uso

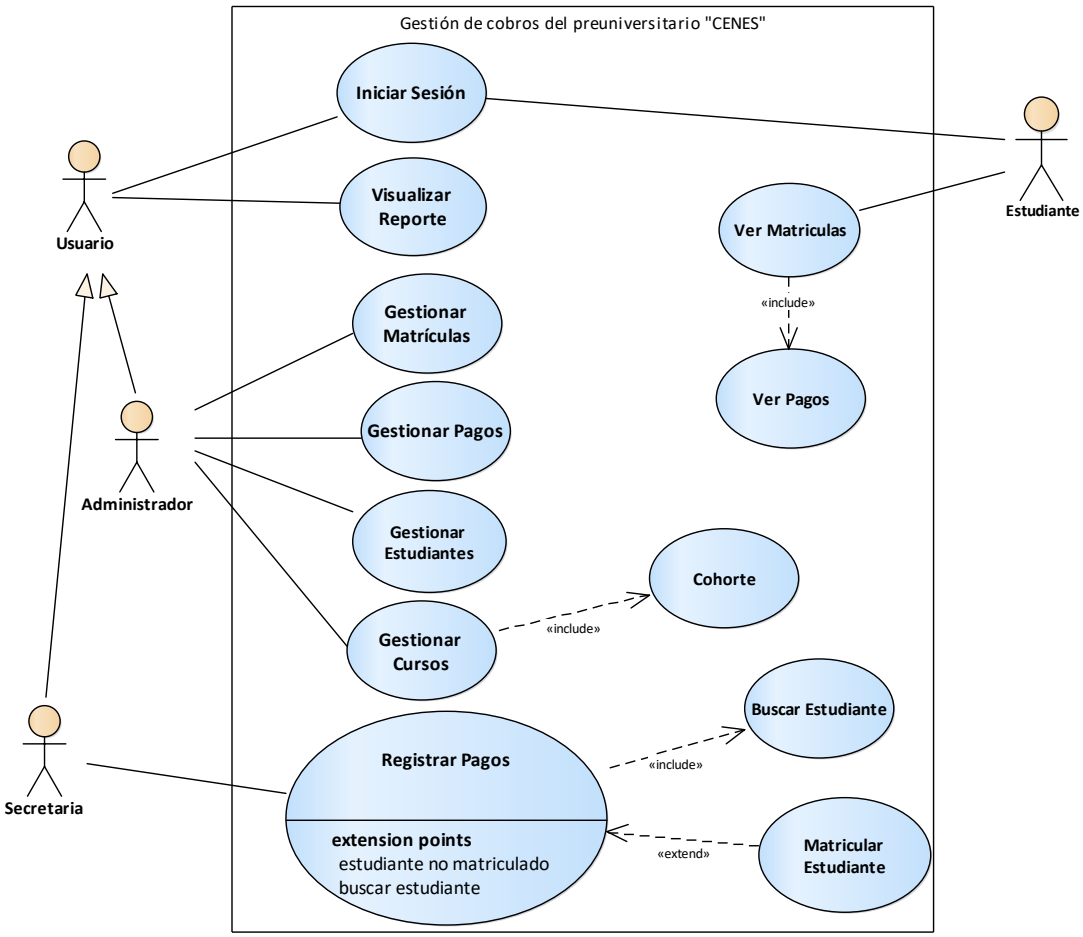


Figura 1 Caso de uso Gestión de cobros

3. Descripción de Actores

Un actor es cualquier entidad externa al sistema modelado que interactúa con él.

1) Administrador

Tabla 1 Descripción del actor Administrador

Actor	Administrador	Identificador: A1
Descripción	El actor administrador es el encargado de la gestión del sistema.	
Referencias	CU1, CU2, CU3, CU4, CU5, CU6, CU7	
Atributos		
Nombre	**Descripción**	**Tipo**
Nombres	Nombre completo del actor	String
Apellidos	Apellido completo del actor	String
Correo	Correo personal de actor	String

2) Secretaria

Tabla 2 Descripción del actor secretaria

Actor	Secretaria	Identificador: A2
Descripción	El actor secretaria es la encargada del registro de los estudiantes y cada uno de sus pagos.	
Referencias	CU1, CU5, CU6, CU7	
Atributos		
Nombre	**Descripción**	**Tipo**
Nombres	Nombre completo del actor	String
Apellidos	Apellido completo del actor	String
Correo	Correo personal de actor	String

4. Especificación de Casos de Uso

En la presente Especificación del caso de uso, se describe la forma en que los actores interactúan con el sistema, listando las funciones o tareas a realizar, los datos de entrada, información que necesita recibir el actor del sistema, información sobre eventos o cambios inesperados, entre otros.

4.1.Iniciar Sesión

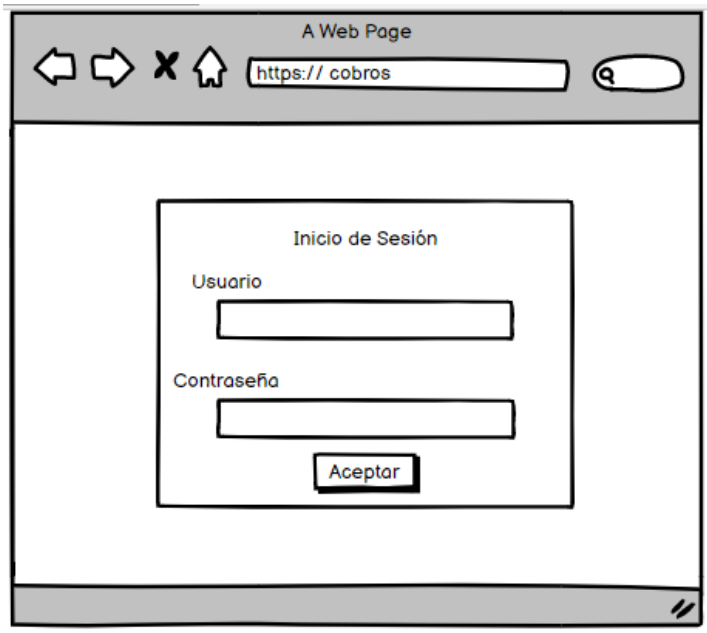


Figura 2 Prototipo de pantalla de inicio de Sesión

Tabla 3 Especificación del caso de uso Iniciar Sesión

Caso de Uso	Iniciar Sesión	**Identificador:** CU01
Actores	**Usuario**	
Tipo	Primario	
Referencias	RF001	
Precondición	El Usuario haya ingresado a la interfaz ingresar al sistema	
Secuencia Normal para ingresar al sistema		
Usuario	**Sistema**	
1. Ingresa los datos: usuario y contraseña y hace clic en el botón “Iniciar Sesión”.	2. Valida la información ingresada.	
	3. Busca un usuario con las credenciales ingresadas	
	4. Da acceso al usuario y muestra la página principal con los permisos de acuerdo al tipo usuario.	
	5. El caso de uso termina.	
Excepciones	**Sistema**	
3. Error credenciales incorrectas	a. Muestra un mensaje de error solicitando que ingrese de nuevo los datos. b. El sistema validará en 3 oportunidades de inicio de sesión y luego se bloquea. b. Regresar al paso 1, hasta cumplir el flujo.	

Postcondición	El usuario inicia sesión con éxito.

4.2.Gestionar Cursos

1) Crear Curso

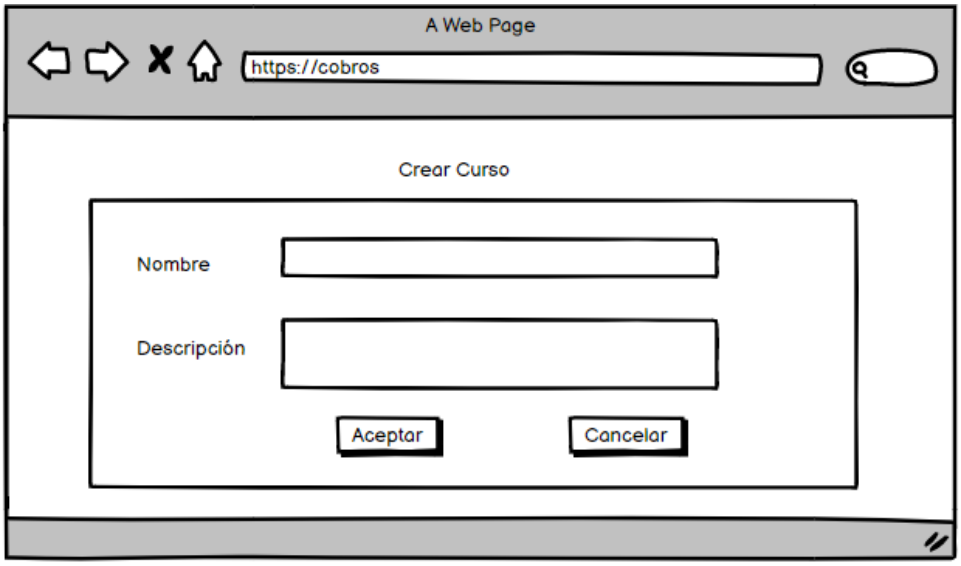


Figura 3 Prototipo de pantalla para crear un curso

Tabla 4 Descripción de caso de uso Gestionar Cursos-Crear

Caso de Uso	Crear Curso	**Identificador:** CU06
Actores	**Administrador**	
Tipo	Primario	
Referencias	RF002	
Descripción	El usuario puede registrar un nuevo curso	
Precondición	Iniciar Sesión Haber seleccionado el botón crear "Nuevo Curso"	
Secuencia Normal para crear un curso		
Administrador	**Sistema**	
1. Ingresa en el formulario presentado los datos: - Nombre del Curso - Descripción 2. Clic en el botón Aceptar		
	3. Valida la información ingresada	
	4. Guarda los datos	
	5. Muestra un mensaje de confirmación	
	6. El caso de uso termina	
Excepciones	**Sistema**	
4. Error al ingresar los datos	a. Muestra un mensaje de error en caso de que algún dato ingresado sea incorrecto. b. Regresar al paso 1, hasta cumplir el flujo.	

Postcondición	Se registra un nuevo curso con éxito.

2) Editar Curso

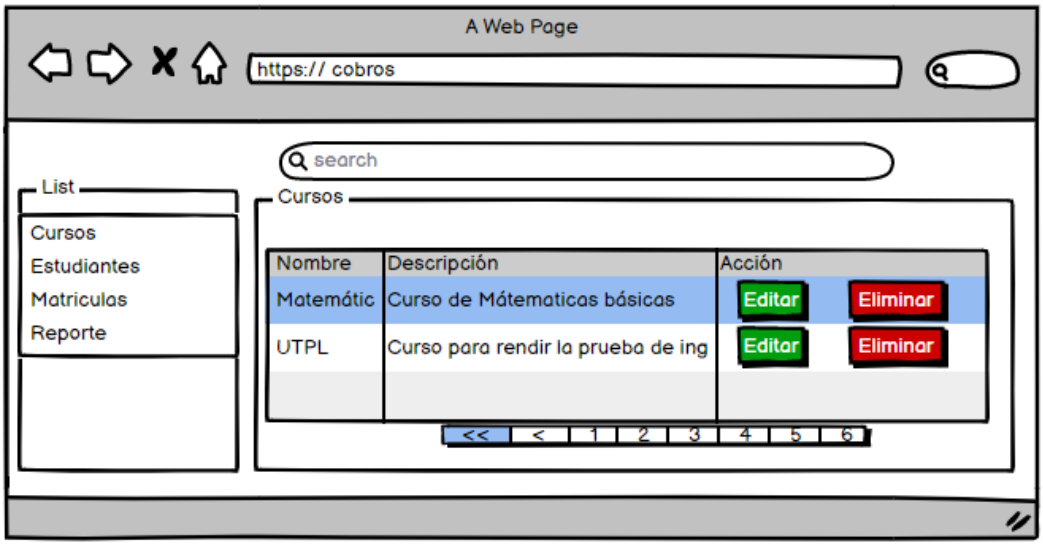


Figura 4 Prototipo de pantalla de listar cursos

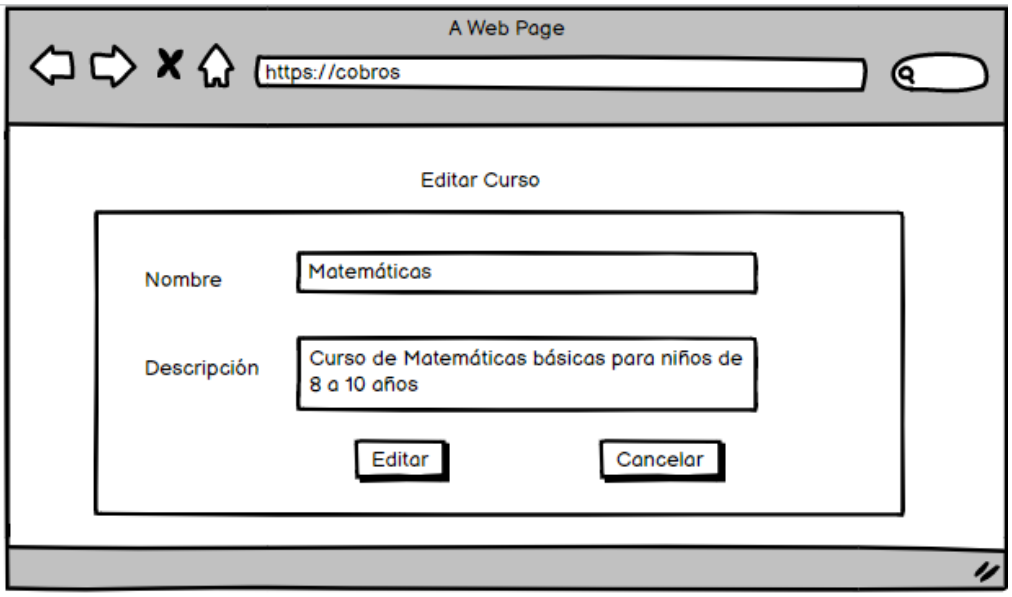


Figura 5 Prototipo de pantalla para editar un curso

Tabla 5 Descripción del caso de uso Gestionar Cursos- Editar

Caso de Uso	Editar Curso	**Identificador:** CU06
Actores	**Administrador**	
Tipo	Secundario	
Referencias	RF002	
Descripción	El usuario pude editar un curso previamente creado	
Precondición	Iniciar Sesión Haber ingresado a la interfaz Ver Cursos El sistema debe tener registrado a menos un curso	
Secuencia Normal para editar un curso		
Administrador	**Sistema**	
	1. Busca todos los cursos registrados	
	2. Carga la lista de cursos en la tabla curso de la interfaz Ver Cursos	
3. Selecciona un curso y da clic en el botón “Editar”.		
	4. Obtiene el id del curso de la tabla Curso 5. Buscar el curso a través del id 6. Carga los datos del curso en los campos de la interfaz crear / editar curso	
7. Cambia los valores en los campos que desea y da clic en el botón “Aceptar”		
	8. Valida la información ingresada 9. Actualiza los datos 10. Muestra un mensaje de confirmación 11. El caso de uso termina	
Excepciones	**Software**	
8. Error al ingresar los datos	a. Muestra un mensaje de error en caso de que algún dato ingresado sea incorrecto. b. Regresar al paso 7, hasta cumplir el flujo.	
Postcondición	Se actualizan los datos de un curso con éxito.	

3) Eliminar Curso

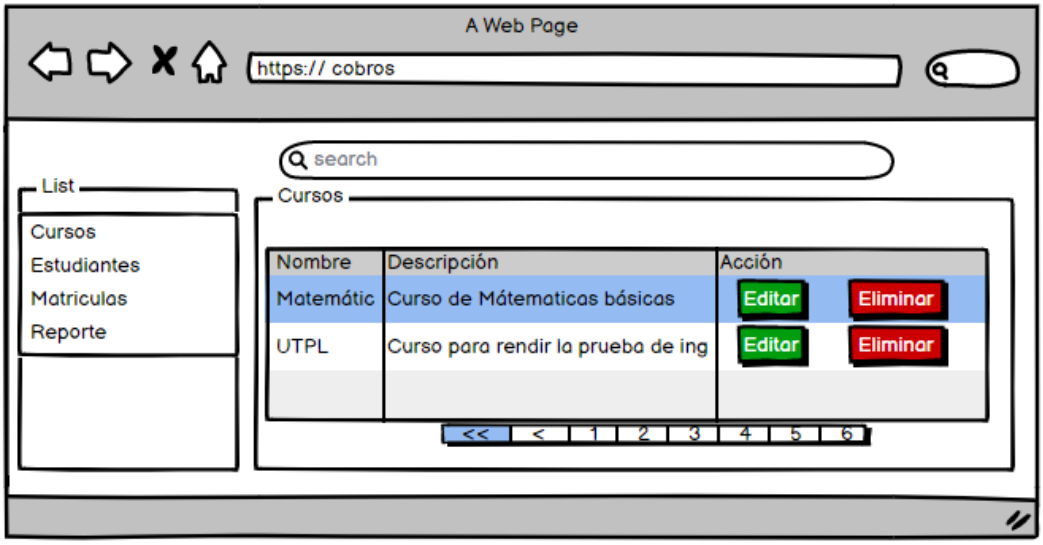


Figura 6 Prototipo de pantalla para listar cursos

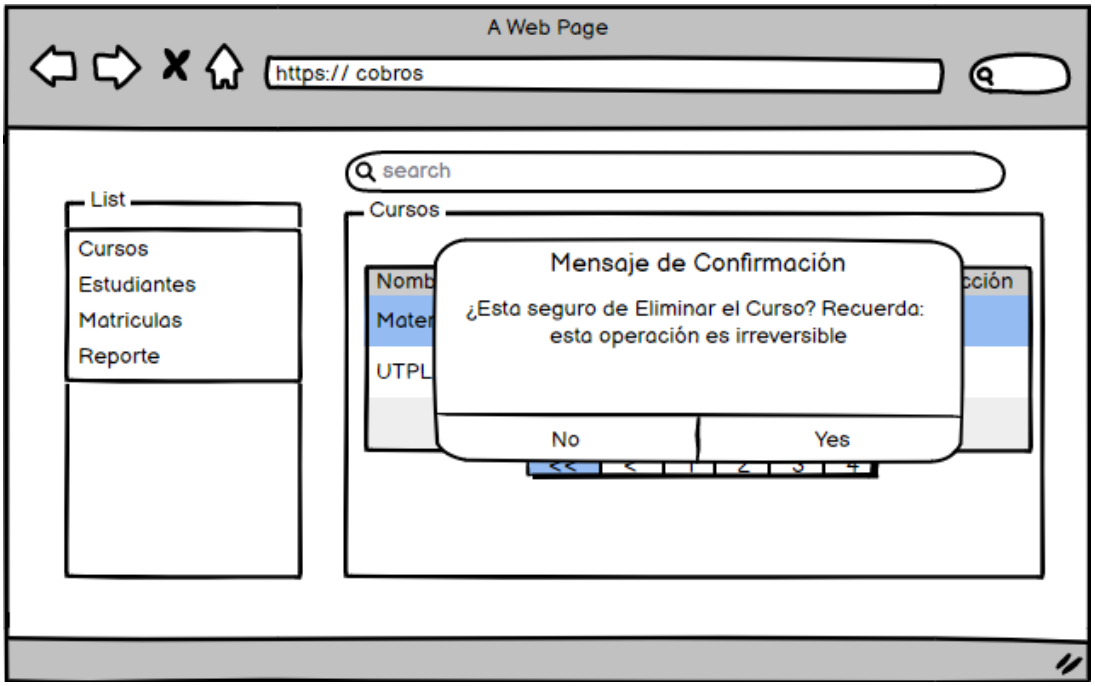


Figura 7 Prototipo de pantalla para eliminar un curso

Tabla 6 Descripción del caso de uso Eliminar Curso

Caso de Uso	Eliminar Curso	**Identificador:** CU06
Actores	**Administrador**	
Tipo	Eliminar	
Referencias	RF002	
Descripción	El usuario pude eliminar/cerrar un curso cuando este haya culminado o por que se creó por error	
Precondición	Iniciar Sesión Haber ingresado a la interfaz Ver Cursos El sistema debe tener registrado a menos un curso.	
Secuencia Normal para Eliminar un curso		
Administrar	**Sistema**	
	1. Busca todos los cursos registrados	
	2. Carga la lista de cursos en la tabla curso de la interfaz Ver Cursos	
3. Selecciona un curso y da clic en el botón “Eliminar”.		
	4. Muestra un mensaje de confirmación	
5. Da clic en “Confirmar”		
	6. Buscar el curso y verifica que no tenga estudiantes matriculados	
	7. Elimina el curso	
	8. Recarga la tabla de cursos	
	9. El sistema muestra un mensaje de confirmación.	
	10. El caso de uso termina.	
Excepciones	**Software**	
Postcondición	Se elimina un curso con éxito.	

4.3.Gestionar Cohorte

1) Crear Cohorte

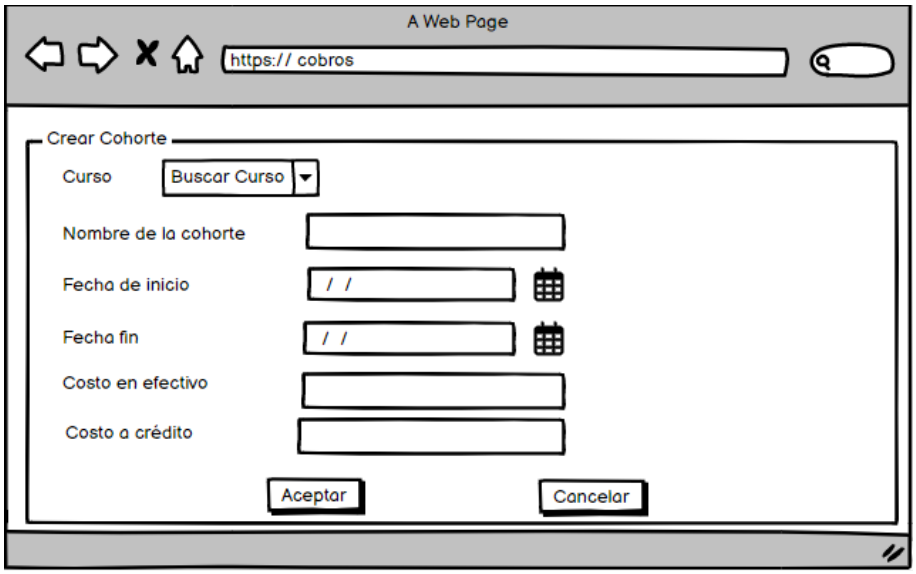


Figura 8 Prototipo de pantalla para Crear

Tabla 7 Descripción del caso de uso Crear Estudiante

Caso de Uso	Crear cohorte	**Identificador:** CU07
Actores	**Administrador**	
Tipo	Primario	
Referencias	RF003	
Precondición	Iniciar Sesión Ingresar en la interfaz de Registrar Cohorte	
Secuencia Normal para crear una Cohorte		
Usuario	**Sistema**	
1. Selecciona el curso en la lista desplegable 2. Ingresa los datos - Nombre de la Cohorte - Fecha de inicio - Fecha Fin - Costo en Efectivo - Costo a crédito 3. Presiona el botón Aceptar		
	4. Valida la información ingresada.	
	5. Guarda los datos	
	6. Muestra un mensaje de confirmación.	
	7. El caso de uso termina.	
Excepciones	**Sistema**	
4. Error al ingresar los datos	a. Muestra un mensaje de error en caso de que algún dato ingresado sea incorrecto. b. Regresar al paso 1, hasta cumplir el flujo.	

Postcondición	Registro de la cohorte con éxito.

2) Editar Cohorte

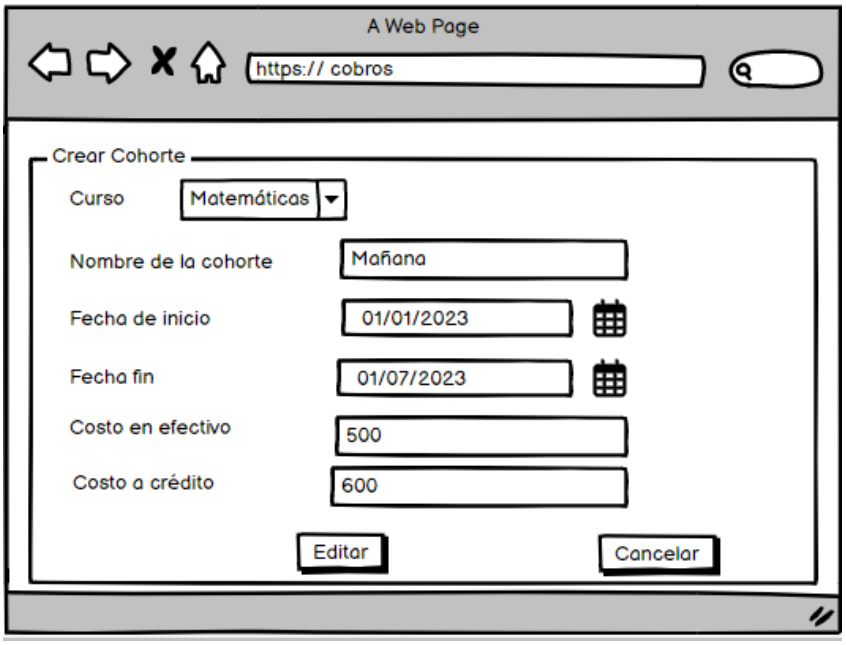


Figura 9 Prototipo de pantalla para editar una cohorte

Tabla 8 Descripción del caso de uso Cohorte

Caso de Uso	Editar Cohorte	**Identificador:** CU07
Actores	**Administrador**	
Tipo	Secundario	
Referencias	RF003	
Descripción	El usuario pude editar una cohorte previamente creado	
Precondición	Iniciar Sesión Haber ingresado a la interfaz Listar Cohorte El sistema debe tener registrado a menos una cohorte	
Secuencia Normal para editar una cohorte		
Administrador	**Sistema**	
	1. Busca todas las cohortes registradas	
	2. Carga la lista de cohortes en la tabla Cohorte de la interfaz Listar Cohorte	
3. Selecciona una cohorte y da clic en el botón “Editar”.		
	4. Obtiene el id de la cohorte de la tabla Cohorte 5. Buscar la cohorte a través del id 6. Carga los datos de la cohorte en los campos de la interfaz crear / editar curso	
7. Cambia los valores en los campos que desea y da clic en el botón “Editar”		
	8. Valida la información ingresada 9. Actualiza los datos 10. Muestra un mensaje de confirmación	

	11. El caso de uso termina
Excepciones	**Software**
8. Error al ingresar los datos	a. Muestra un mensaje de error en caso de que algún dato ingresado sea incorrecto. b. Regresar al paso 7, hasta cumplir el flujo.
Postcondición	Se actualizan los datos de una cohorte con éxito.

3) Eliminar Cohorte

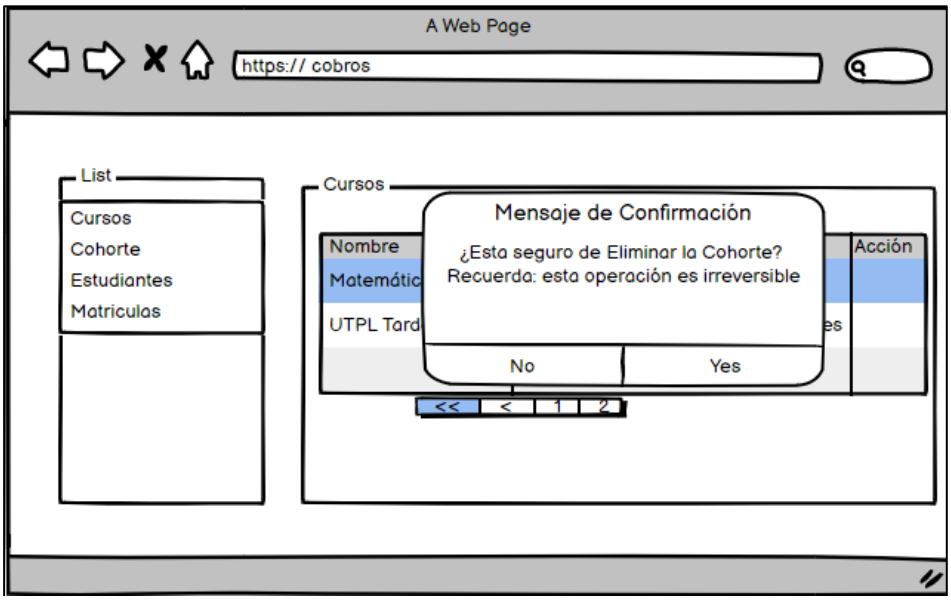


Figura 10 Prototipo de pantalla para eliminar una cohorte

Tabla 9 Descripción del caso de uso Cohorte

Caso de Uso	Eliminar Cohorte	**Identificador:** CU07
Actores	**Administrador**	
Tipo	Secundario	
Referencias	RF003	
Descripción	El usuario puede eliminar una cohorte.	
Precondición	Iniciar Sesión Haber ingresado a la interfaz Listar Cohorte El sistema debe tener registrado a menos una cohorte.	
Secuencia Normal para Eliminar una cohorte		
Administrar	**Sistema**	
	1. Busca todas las cohortes registradas	
	2. Carga la lista de cohortes en la tabla Cohorte de la interfaz Listar Cohorte	
3. Selecciona un curso y da clic en el botón “Eliminar”.		
	4. Muestra un mensaje de confirmación	
5. Da clic en “Confirmar”		
	6. Elimina la cohorte	
	7. Recarga la tabla de cohorte	
	8. El sistema muestra un mensaje de confirmación.	
	9. El caso de uso termina.	

Excepciones	Software
Postcondición	Se elimina una cohorte con éxito.

4.4.Gestionar estudiantes

1) Crear Estudiante

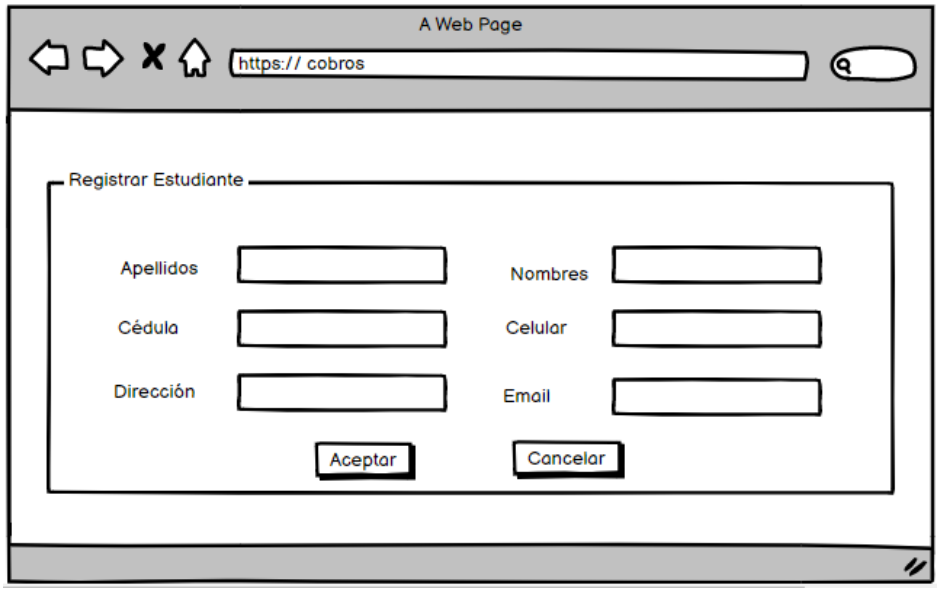


Figura 11 Prototipo de pantalla para crear un estudiante

Tabla 10 Descripción del caso de uso Gestionar Estudiante

Caso de Uso	Crear Estudiante	**Identificador:** CU05
Actores	**Administrador, secretaria**	
Tipo	Primario	
Referencias	RF004	
Precondición	Iniciar Sesión Ingresar en la interfaz de Registrar Estudiante	
Secuencia Normal para registrar un estudiante		
Usuario	**Sistema**	
1. Ingresa en el formulario presentado los datos Datos del Estudiante: - Apellidos - Nombres. - Cedula de identidad. - Celular - Dirección - Correo electrónico		
2. Da clic en el botón “Crear”		
	3. Valida la información ingresada.	
	4. Guarda los datos	
	5. Muestra un mensaje de confirmación.	
	6. El caso de uso termina.	
Excepciones	**Sistema**	
3. Error al ingresar los datos	a. Muestra un mensaje de error en caso de que algún dato ingresado sea incorrecto. b. Regresar al paso 1, hasta cumplir el flujo.	
Postcondición	Registro del estudiante con éxito.	

2) Editar Estudiante

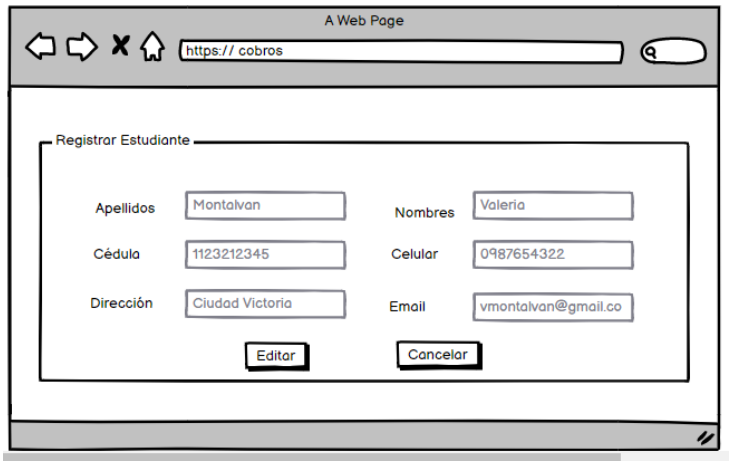


Figura 12 Prototipo de pantalla para editar un estudiante

Tabla 11 Detalle del caso de uso Gestionar Estudiante Editar

Caso de Uso	Editar Estudiante	**Identificador:** CU05
Actores	**Administrador**	
Tipo	Secundario	
Referencias	RF005	
Descripción	El usuario pude editar un estudiante que se encuentre registrado.	
Precondición	Iniciar Sesión Ingresar en la interfaz de Listar Estudiante El sistema debe tener registrado al menos un estudiante	
Secuencia Normal para editar un estudiante		
Administrador	**Sistema**	
1. Da clic en el botón “Editar”		
	2. Obtiene el id del estudiante de la tabla Estudiante 3. Buscar el estudiante a través del id 4. Carga los datos del estudiante en los campos de la interfaz crear / editar estudiante	
5. Cambia los valores en los campos que desea y da clic en el botón “Editar”		
	6. Valida la información ingresada 7. Actualiza los datos 8. Muestra un mensaje de confirmación 9. El caso de uso termina	
Excepciones	**Software**	
6. Error al ingresar los datos	a. Muestra un mensaje de error en caso de que algún dato ingresado sea incorrecto. b. Regresar al paso 5, hasta cumplir el flujo.	
Postcondición	Se actualizan los datos de un estudiante con éxito.	

4.5.Gestionar Matrícula

1) Crear Matrícula

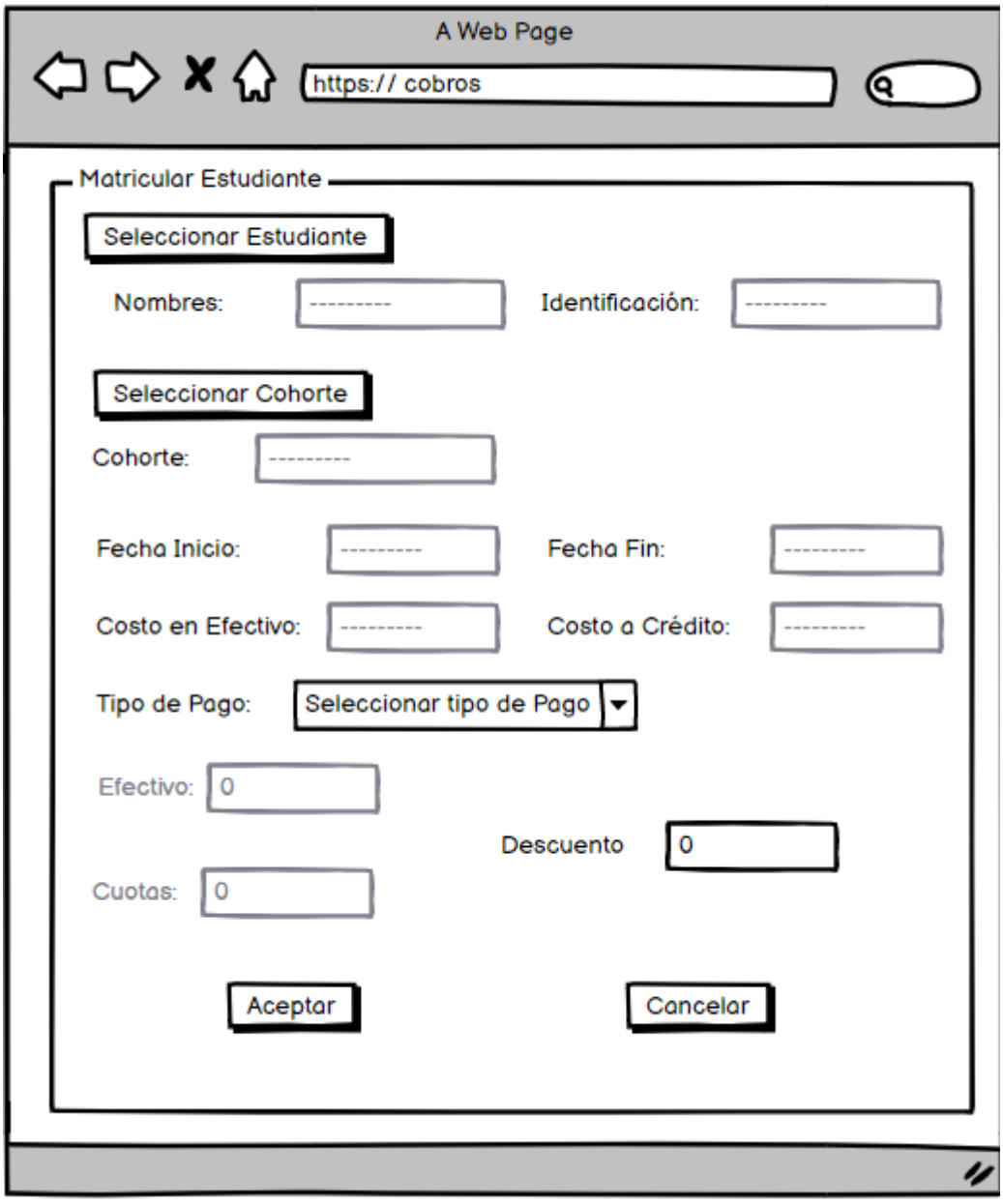


Figura 13 Prototipo de pantalla para crear una matrícula

Tabla 12 Descripción del caso de uso Matricular-Crear Matrícula

Caso de Uso	Crear Matrícula	**Identificador:** CU03
Actores	**Administrador, secretaria**	
Tipo	Primario	
Referencias	RF006	
Precondición	Iniciar Sesión Ingresar en la interfaz de Crear Matrícula	
Secuencia Normal para registrar una matrícula		
Usuario	**Sistema**	
15. Selecciona el botón “Estudiante” 16. Seleccionar/Buscar el estudiante		
	17. Muestra datos del estudiante seleccionado	
18. Selecciona el botón “Cohorte” 19. Seleccionar/Buscar la Cohorte a la que se va a matricular.		
	20. Muestra los datos de la Cohorte seleccionada	
21. Seleccionar el tipo de pago “Efectivo”		
	22. Activa el campo Efectivo	
23. Ingresa el valor del pago		
24. Ingresar el valor de descuento, en caso de haberlo o si no dejar el valor por defecto y da clic en “Aceptar”		
	25. Valida la información ingresada.	
	26. Guarda los datos	
	27. Muestra un mensaje de confirmación.	
	28. El caso de uso termina.	
Secuencia Alterna		
Usuario	**Sistema**	
7.Selecciona tipo de pago “Crédito”	7.1. Activa el campo Cuotas	
7.2. Ingresa cantidad de cuotas Vuelve al paso 9		
Excepciones	**Sistema**	
3. Error al ingresar los datos	a. Muestra un mensaje de error en caso de que algún dato ingresado sea incorrecto. b. Regresar al paso 1, hasta cumplir el flujo.	
Postcondición	Registro de la matrícula con éxito.	

2) Eliminar Matrícula

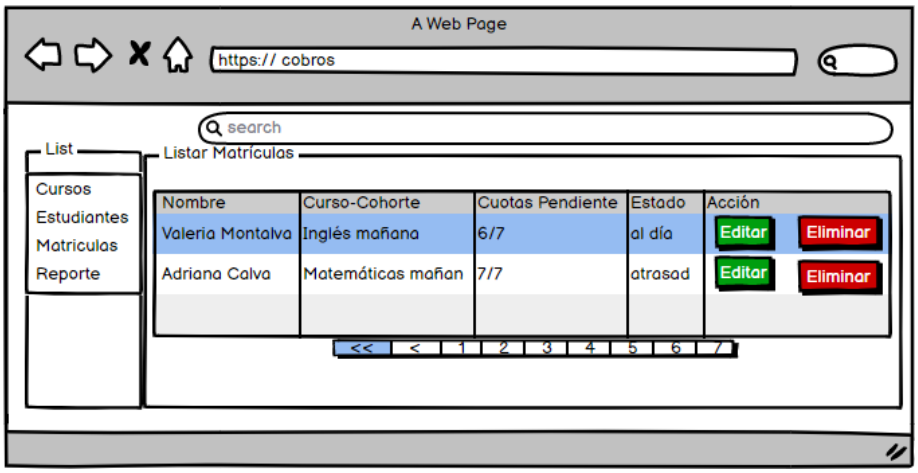


Figura 14 Prototipo de pantalla para listar matrículas

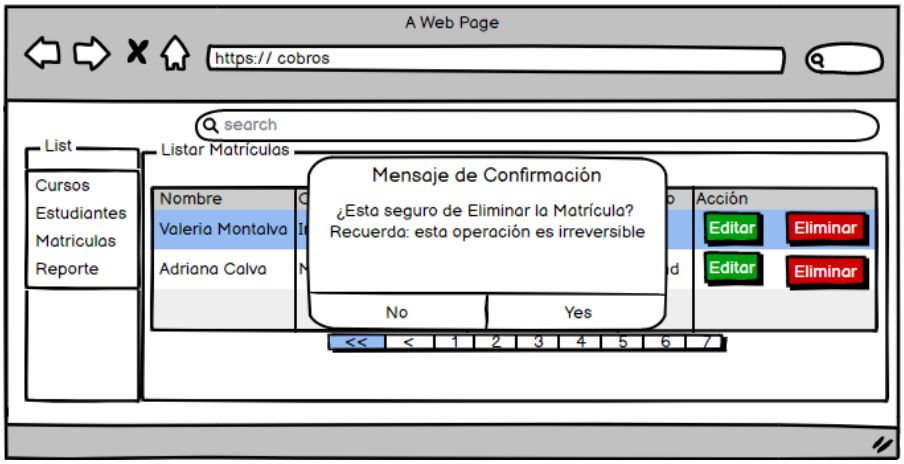


Figura 15 Prototipo de pantalla para eliminar una matrícula

Tabla 13 Descripción del caso de uso Matricular-Eliminar matrícula

Caso de Uso	Eliminar Matrícula	**Identificador:** CU03
Actores	**Administrador**	
Tipo	Secundario	
Referencias	RF006	
Descripción	El usuario puede eliminar una matrícula.	
Precondición	Iniciar Sesión Haber ingresado a la interfaz Listar Matrículas El sistema debe tener registrado al menos una matrícula.	
Secuencia Normal para Eliminar una matrícula		
Administrar	**Sistema**	
	1. Busca todas las matrículas registradas	
	2. Carga la lista de matrículas en la tabla Matrículas de la interfaz Listar Matrículas	
3. Selecciona una matrícula y da clic en el botón “Eliminar”.		
	4. Muestra un mensaje de confirmación	
5. Da clic en “Confirmar”		
	6. Elimina el curso	
	7. Recarga la tabla de matrículas	
	8. El sistema muestra un mensaje de confirmación.	
	9. El caso de uso termina.	
Excepciones	**Software**	
Postcondición	Se elimina una matrícula con éxito.	

4.6.Gestionar Pagos

1) Registrar pagos

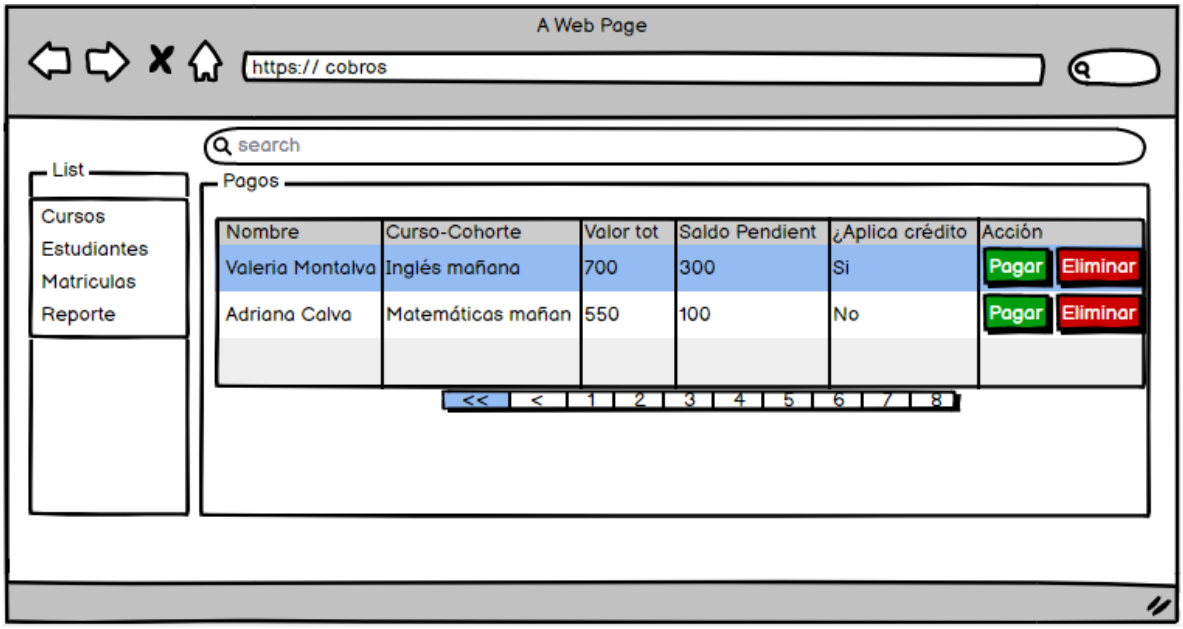


Figura 16 Prototipo de pantalla listar matrículas

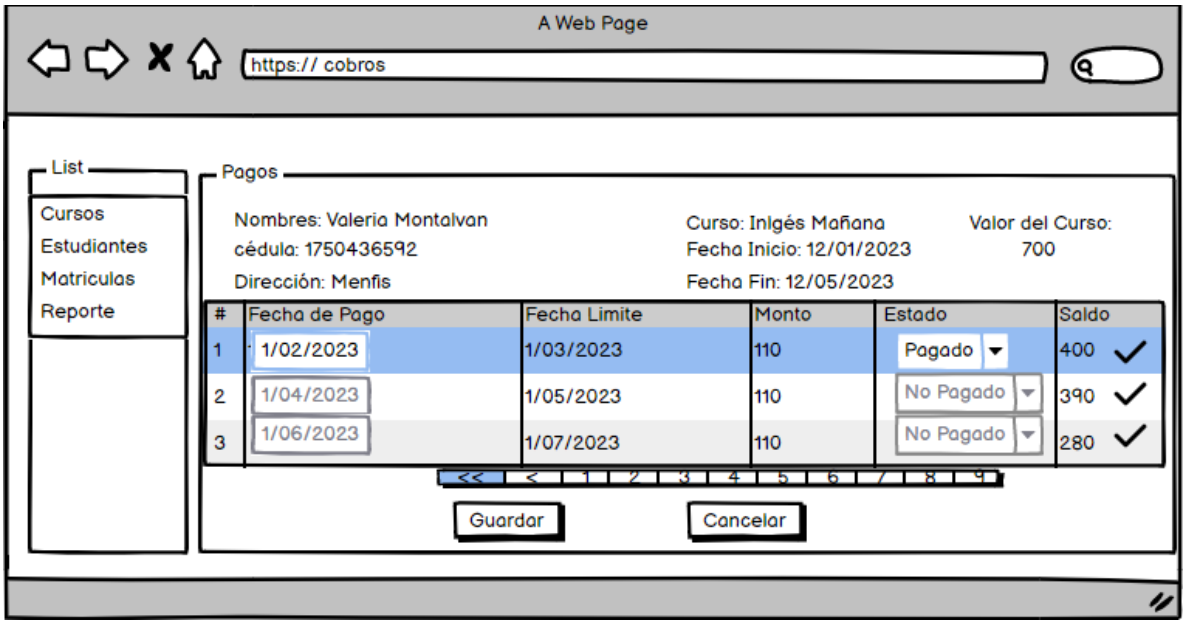


Figura 17 Prototipo de pantalla registrar pago

Tabla 14 Descripción del caso de uso Gestionar Pago-Registrar

Caso de Uso	Registrar pagos	**Identificador:** CU04
Actores	**Administrador, secretaria**	
Tipo	Primario	
Referencias	RF007	
Precondición	Iniciar Sesión Ingresar a la interfaz Registrar Pagos Al menos un pago registrado	
Secuencia Normal para registrar un pago		
Usuario	**Sistema**	
	12. Busca todos los estudiantes matriculados	
	13. Carga la lista de estudiante en la tabla "Pagos", de la interfaz Listar Pagos	
14. Selecciona un estudiante y da clic en el botón "Pagar".		
	15. Muestra la interfaz registrar Pago	
16. Ingresar la fecha del pago 17. Cambiar el estado de pago a "PAGADO" 18. Clic en Guardar		
	19. Valida la acción realizada.	
	20. Guarda los datos	
	21. Muestra un mensaje de confirmación.	
	22. El caso de uso termina.	
Excepciones	**Sistema**	
8. Error al ingresar los datos	a. Muestra un mensaje de error en caso de que algún dato ingresado sea incorrecto. b. Regresar al paso 5, hasta cumplir el flujo.	
Postcondición	Se registra el pago con éxito.	

2) Editar Pago

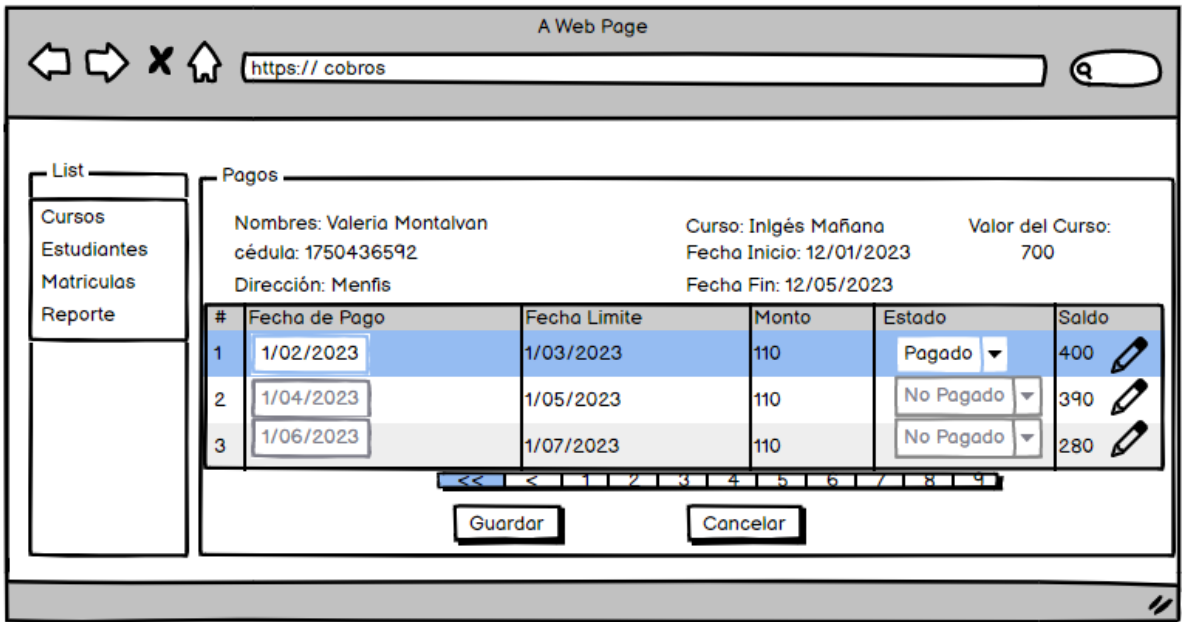


Figura 18 Prototipo de pantalla para editar un pago

Tabla 15 Descripción del caso de uso Gestionar Pagos-Editar Pago

Caso de Uso	Editar Pago	**Identificador:** CU04
Actores	**Usuario**	
Tipo	Secundario	
Referencias	RF008	
Descripción	El usuario pude editar un pago previamente registrado	
Precondición	Iniciar Sesión Ingresar a la interfaz Pagos El usuario debe tener registrado al menos un pago.	
Secuencia Normal para editar un pago		
Usuario	**Sistema**	
1. Presionar el botón “Editar”		
	2. Activa la casilla del pago que se pueden editar	
3. Editar el valor del campo		
	4. Valida los datos 5. Actualiza los datos 6. Muestra un mensaje de confirmación 7. El caso de uso termina	
Excepciones	**Software**	
4. Error al validar datos	a. Muestra un mensaje de error en caso de que el dato ingresado no exista b. Regresar al paso 3, hasta cumplir el flujo.	
Postcondición	Se edita el pago con éxito.	

4.7.Visualizar Reportes

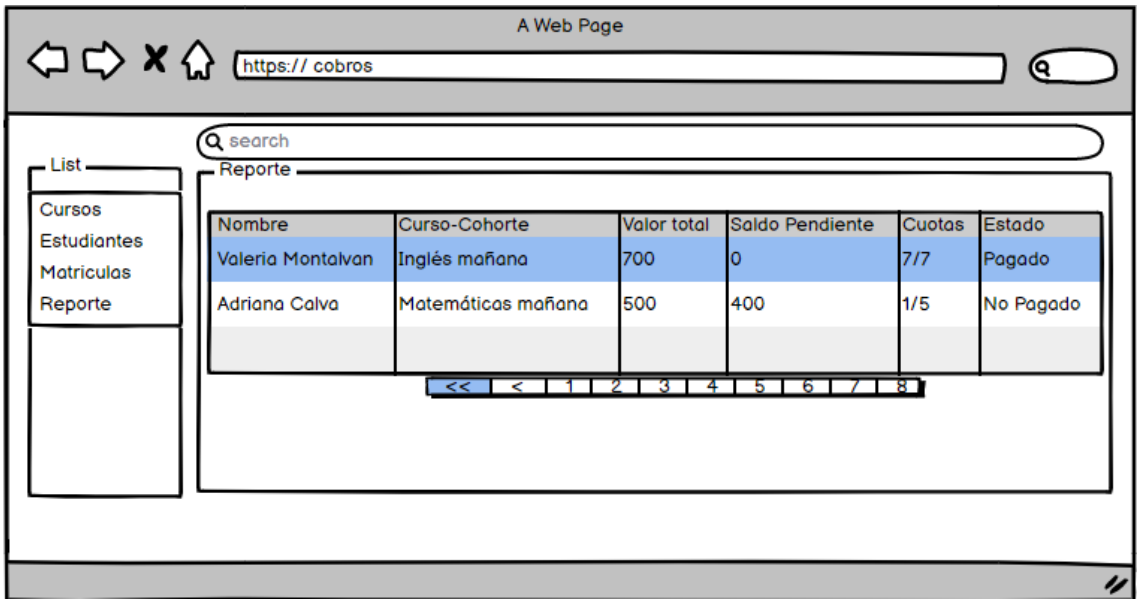


Figura 19 Prototipo de pantalla para visualizar reporte

Tabla 16 Descripción del caso de uso para Reporte
)

Caso de Uso	Ver Reportes	**Identificador:** CU02
Actores	**Administrador, secretaria**	
Tipo	Secundario	
Referencias	RF009	
Descripción	El usuario pude ver los reportes de pago de los estudiantes registrados los cuales se muestran en una tabla que contiene los siguientes datos: Nombres, Curso, Valor del Curso, Saldo Pendiente, Cuotas Pagadas y Estado del Pago, además podrá buscar por estado de pago, Curso o Estudiante.	
Precondición	Iniciar Sesión Ingresa a la interfaz Reportes El sistema debe tener registrado por lo menos un estudiante matriculado.	
Secuencia Normal para visualizar un reporte		
Usuario		**Sistema**
		1. Busca todos los estudiantes matriculados
		2. Carga la lista de estudiante en la tabla “Reporte”, de la interfaz Reportes
3. Ingresa en la barra de búsqueda el dato que desee buscar y presiona el botón “Buscar”		
		4. Busca el dato ingresado

	5. En la interfaz el sistema mostrará una tabla con todos los estudiantes que contengan el dato que fue ingresado en la búsqueda
6. Visualiza los estudiantes que tiene relación con el dato ingresado	
	7. El caso de uso termina.
Excepciones	**Sistema**
4.Error de Búsqueda	a. Muestra un mensaje de error en caso de que el dato ingresado no se haya encontrado b. Regresar al paso 3, hasta cumplir el flujo.
Postcondición	Se genera el reporte de los pagos realizados por los estudiantes.

4.8.Visualizar Matrículas por parte del estudiante

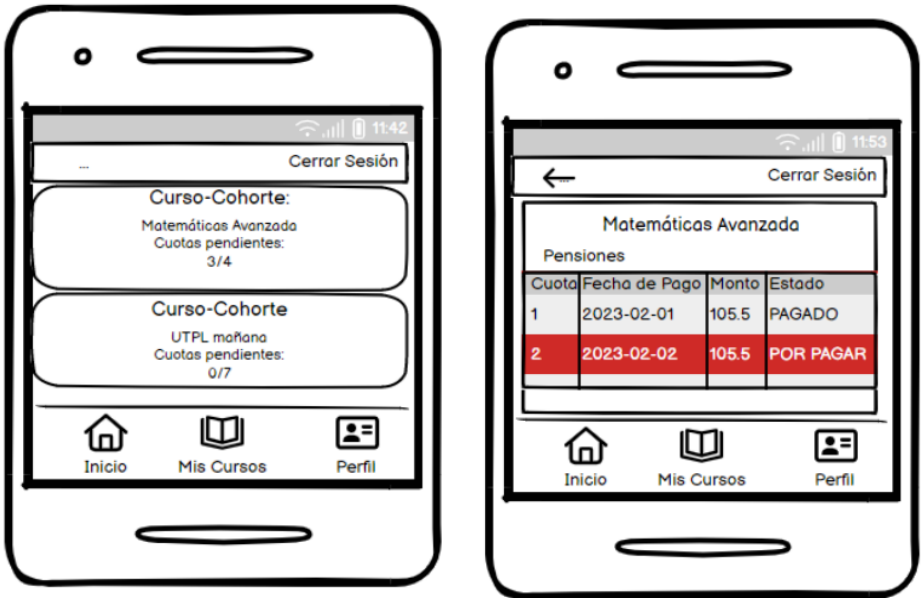


Figura 20 Prototipo de pantalla para visualizar matrículas y pensiones

Tabla 17 Descripción del caso de uso para visualizar matrículas

Caso de Uso	Visualizar Matrículas	**Identificador:** CU09
Actores	**Estudiante**	
Tipo	Secundario	
Referencias	RF010	
Descripción	El usuario pude ver los cursos en los que se encuentra matriculado, además cada curso contiene información sobre los pagos.	
Precondición	Iniciar Sesión Ingresa a la interfaz Cursos El estudiante debe estar matriculado	
Secuencia Normal para visualizar matrículas		
Usuario	**Sistema**	
	1. Busca todos los cursos donde se encuentre matriculado el estudiante que inicio sesión	
	2. Carga la lista de cursos en la tabla “Cursos”, de la interfaz Mis Cursos	
3. Visualiza los cursos donde se encuentra matriculado		
4. Selecciona un curso		
	5. Obtiene el id del curso **6.** Buscar el curso y sus pagos a través del id **7.** Carga los datos de los pagos en los campos de la interfaz Pagos	

8. Visualiza las matrículas con sus pagos	
	9. El caso de uso termina.
Excepciones	**Sistema**
4.Error de Búsqueda	a. Muestra un mensaje de error en caso de que el dato ingresado no se haya encontrado b. Regresar al paso 3, hasta cumplir el flujo.
Postcondición	El estudiante visualiza las matrículas y pagos.

Diagramas de Robustez

Desarrollo de aplicativo web para la gestión de cobro de mensualidades a estudiantes del preuniversitario "CENES"

Contenido:

1. Diagramas de Robustez

1.1. Diagrama de Robustez para Iniciar Sesión

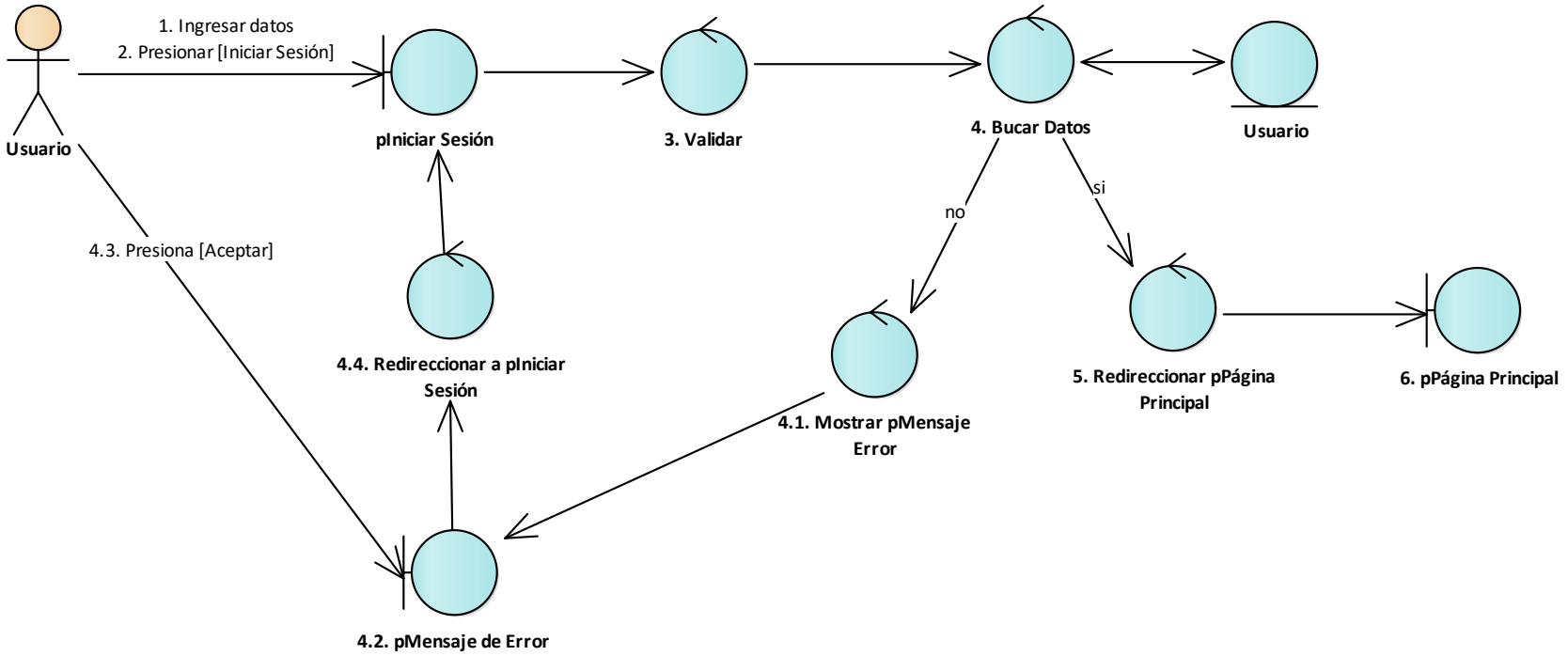


Figura 1 Diagrama de Robustez Inicio de Sesión

1.2. Diagrama de Robustez Crear Curso

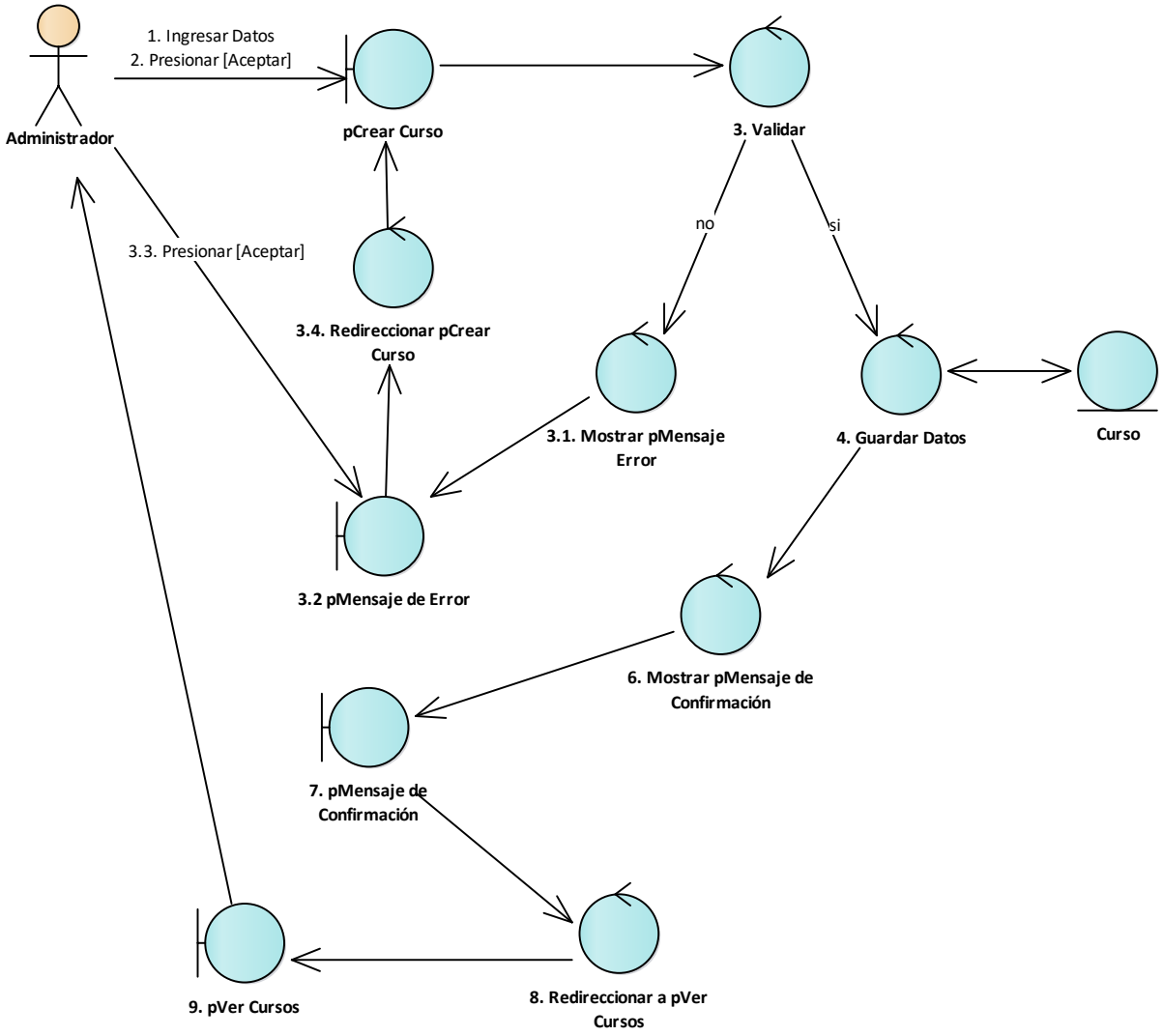


Figura 2 Diagrama de Robustez Registrar Curso

1.3. Diagrama de Robustez Editar Curso

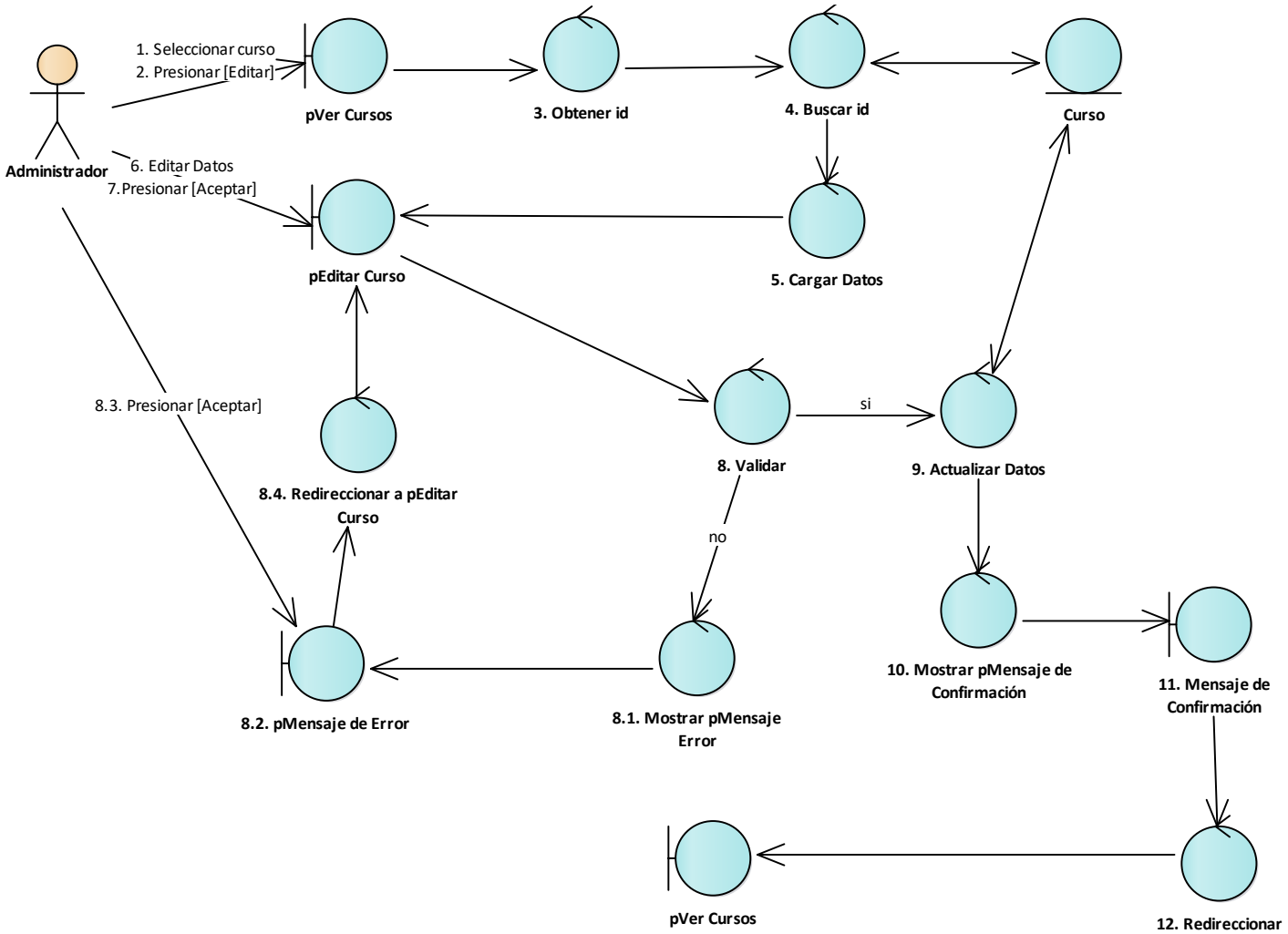


Figura 3 Diagrama de Robustez Editar Curso

1.4. Diagrama de Robustez Buscar Curso

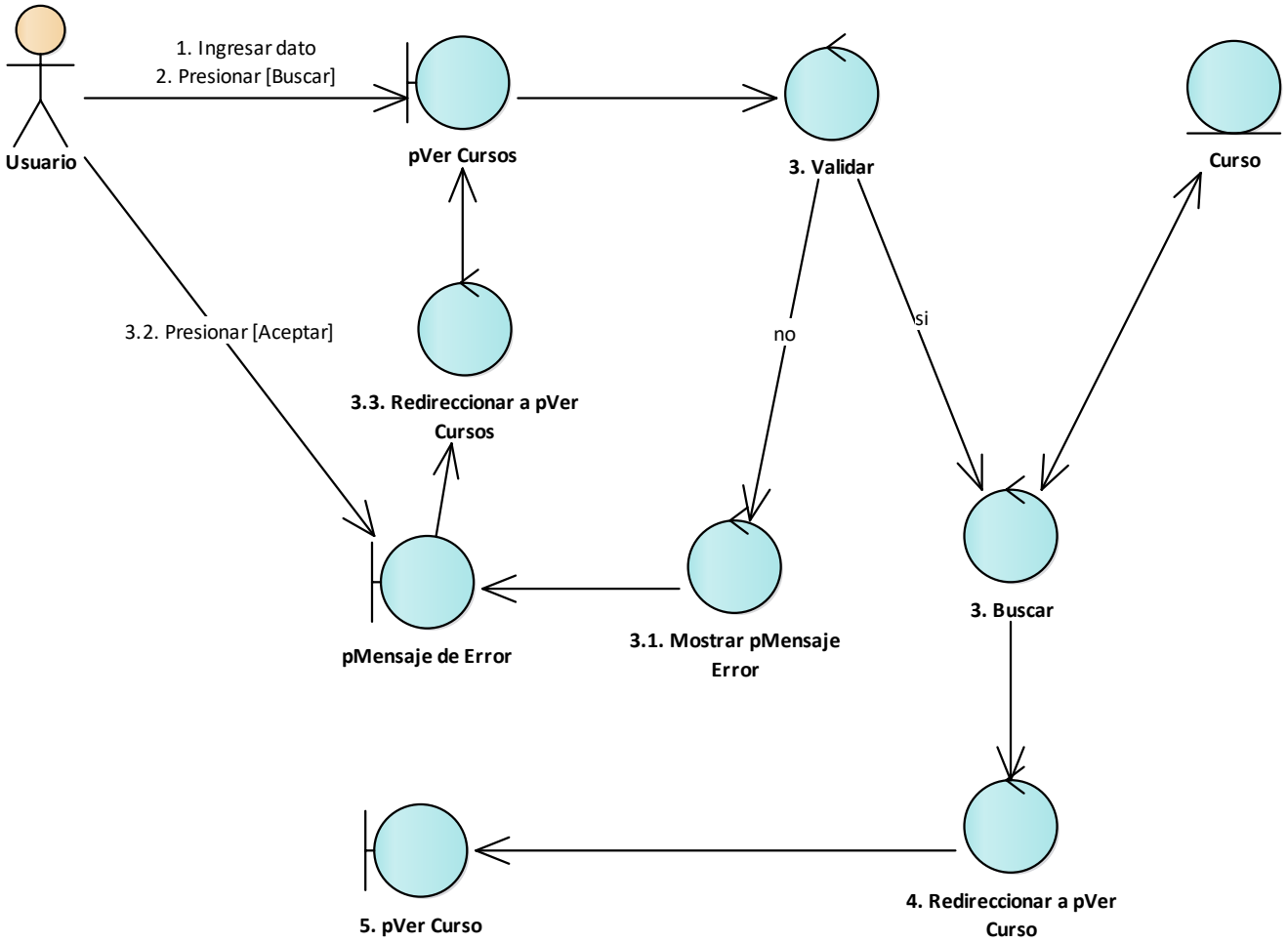


Figura 4 Diagrama de Robustez Buscar Curso

1.5. Diagrama de Robustez Eliminar Curso

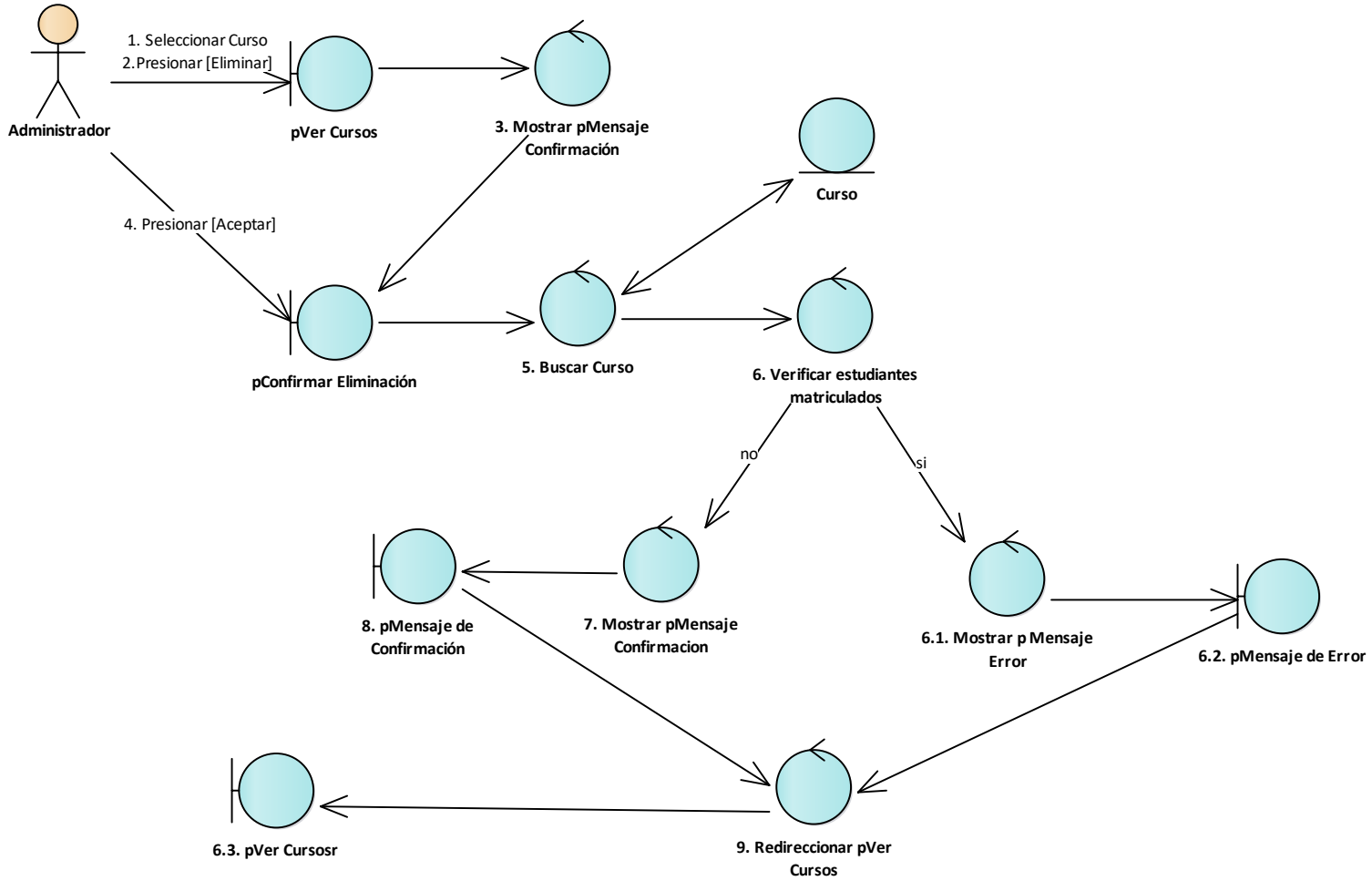


Figura 5 Diagrama de Robustez Eliminar Curso

1.6. Diagrama de Robustez Crear Estudiante

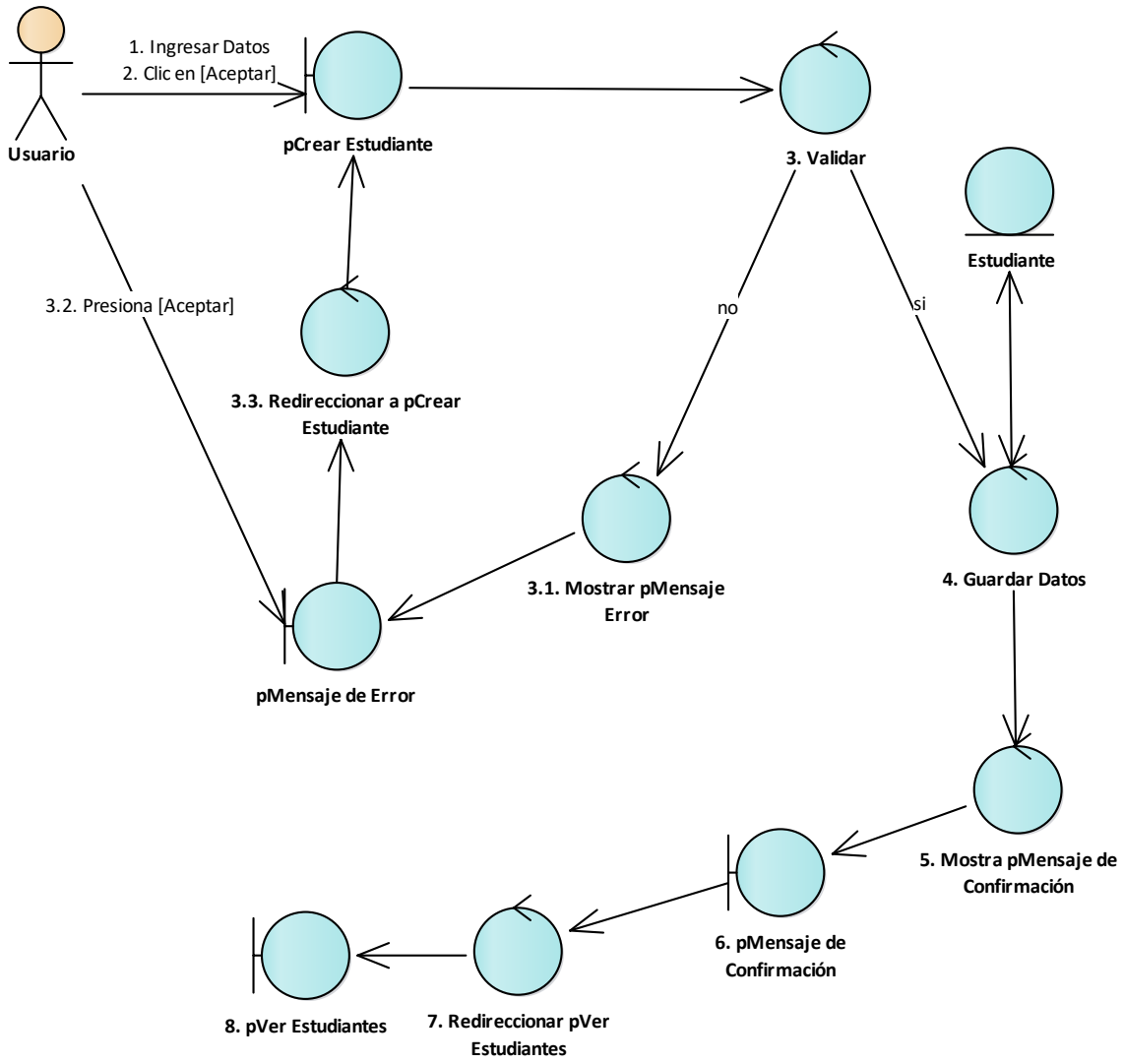


Figura 6 Diagrama de Robustez Crear Estudiante

1.7. Diagrama de Robustez Editar Estudiante

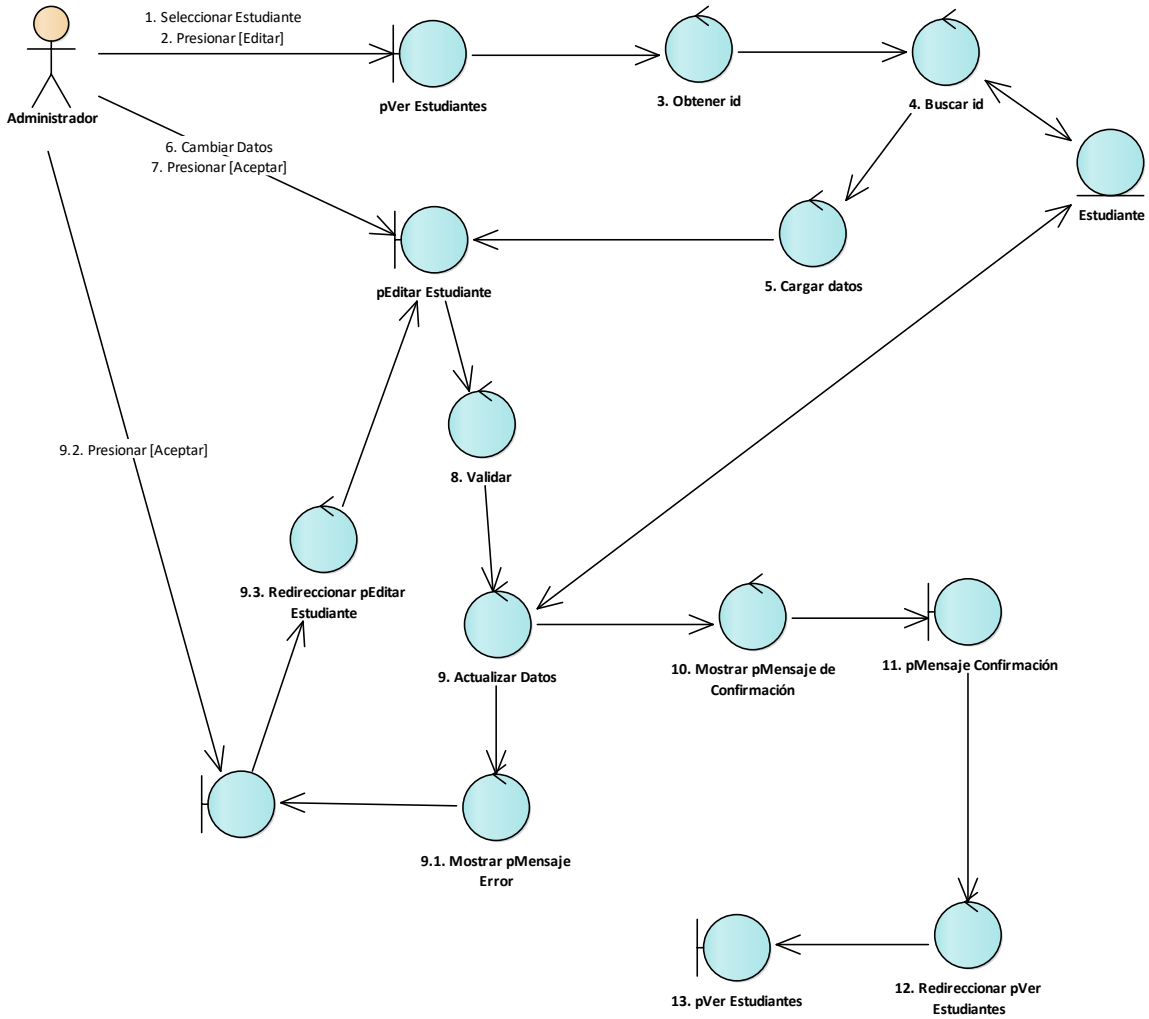


Figura 7 Diagrama de Robustez Editar Estudiante

1.8. Diagrama de Robustez Buscar Estudiante

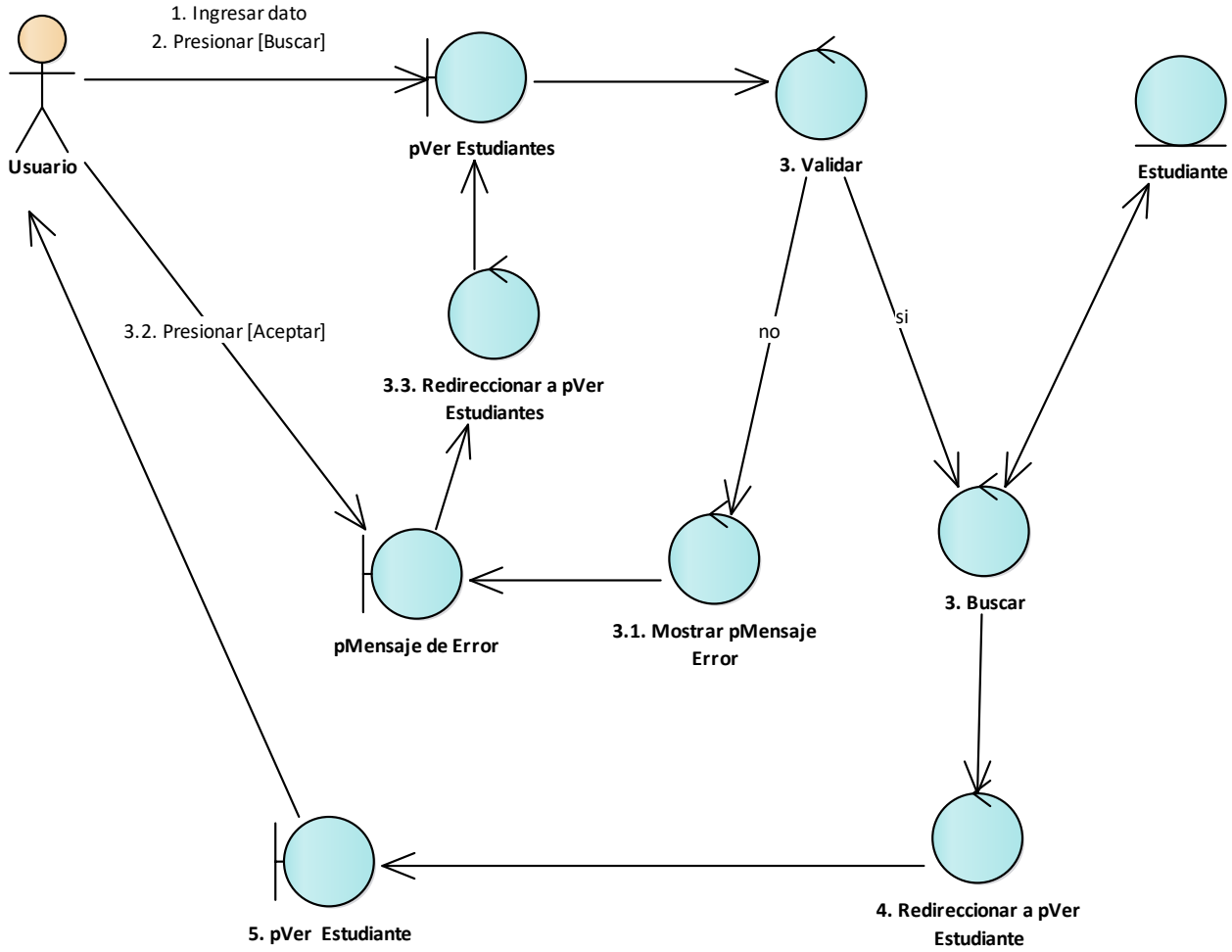


Figura 8 Diagrama de Robustez Buscar Estudiante

1.9. Diagrama de Robustez Registrar Pago

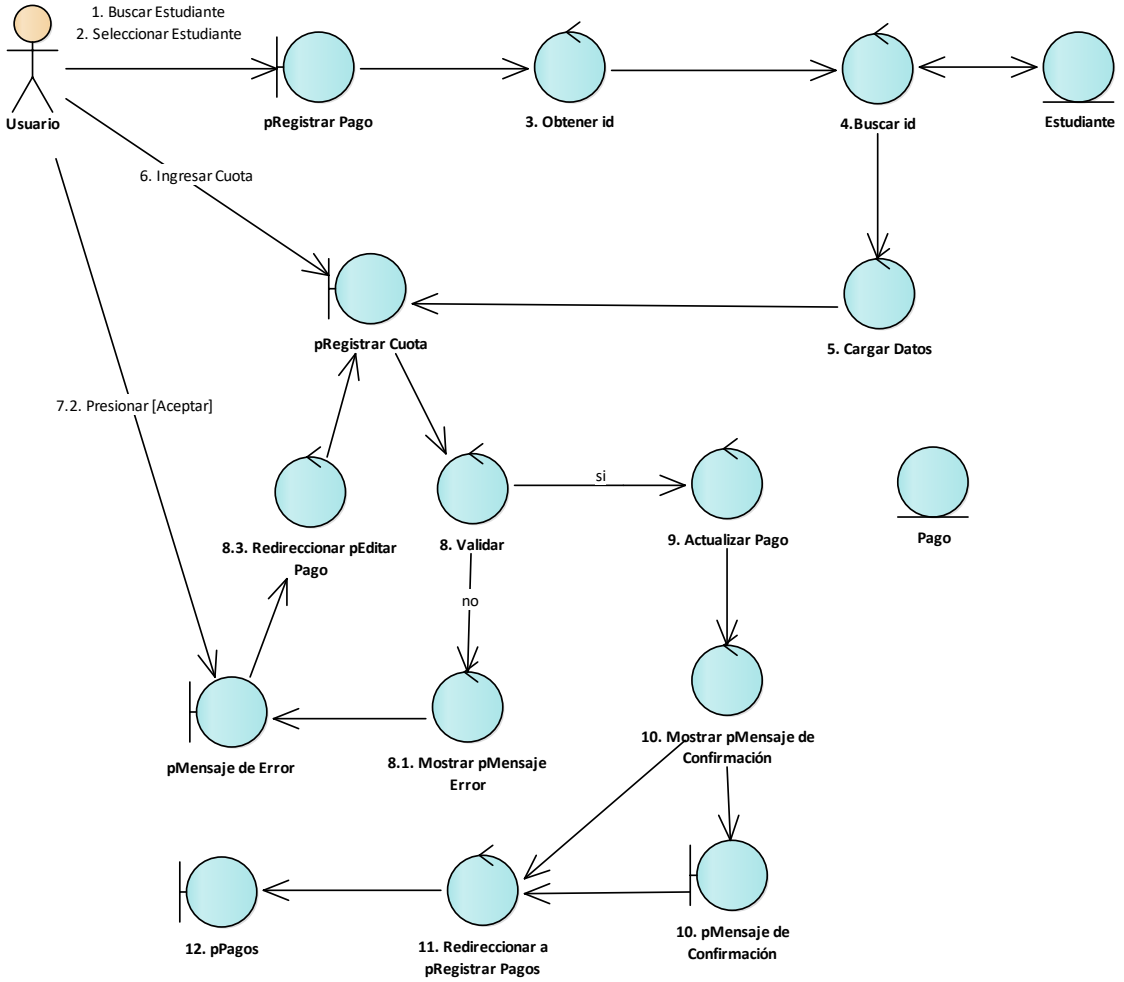


Figura 9 Diagrama de Robustez Registrar Pago

1.10. Diagrama de Robustez Buscar Pago

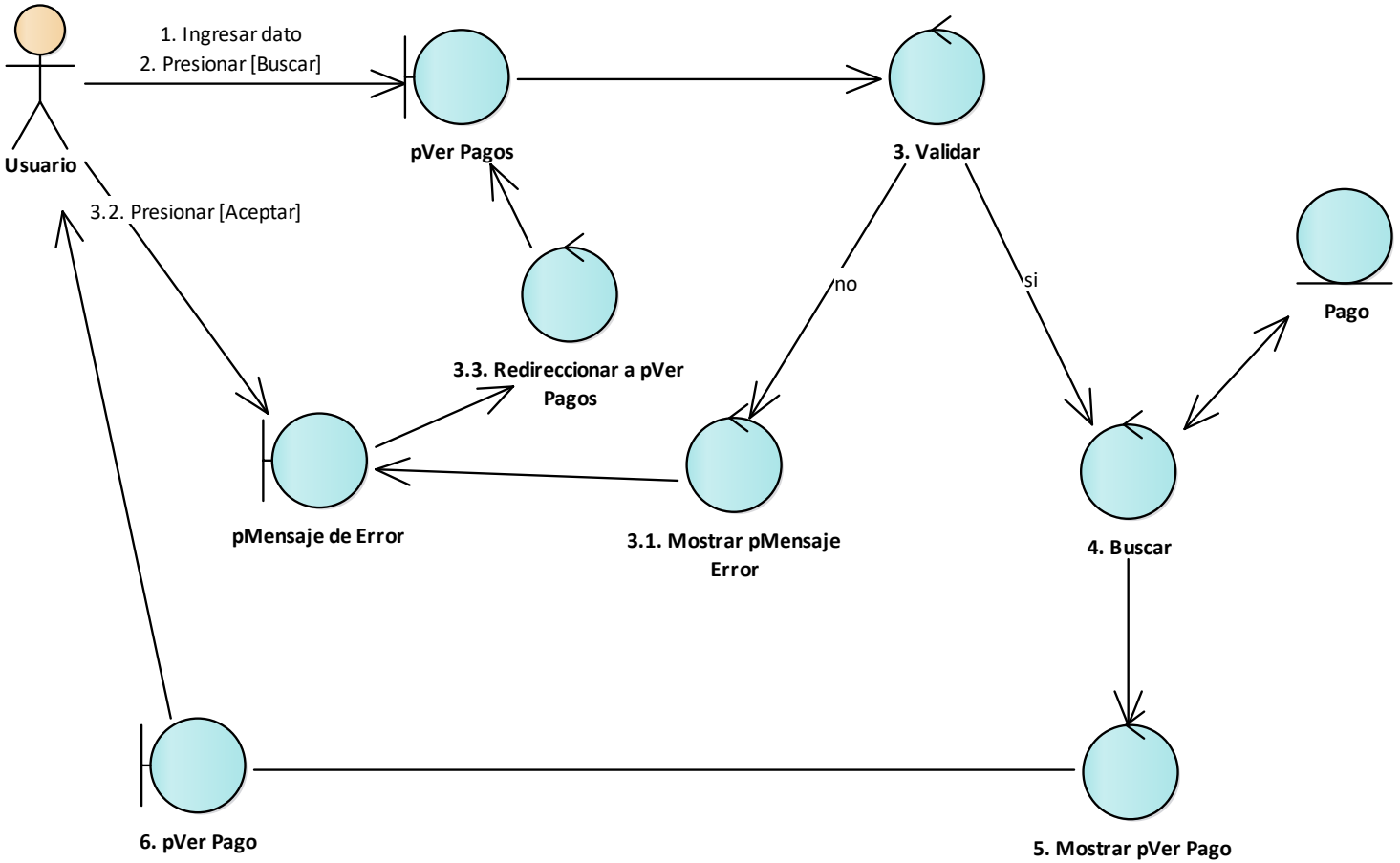


Figura 10 Diagrama de robustez Buscar Pago

1.11. Diagrama de Robustez Ver Reporte

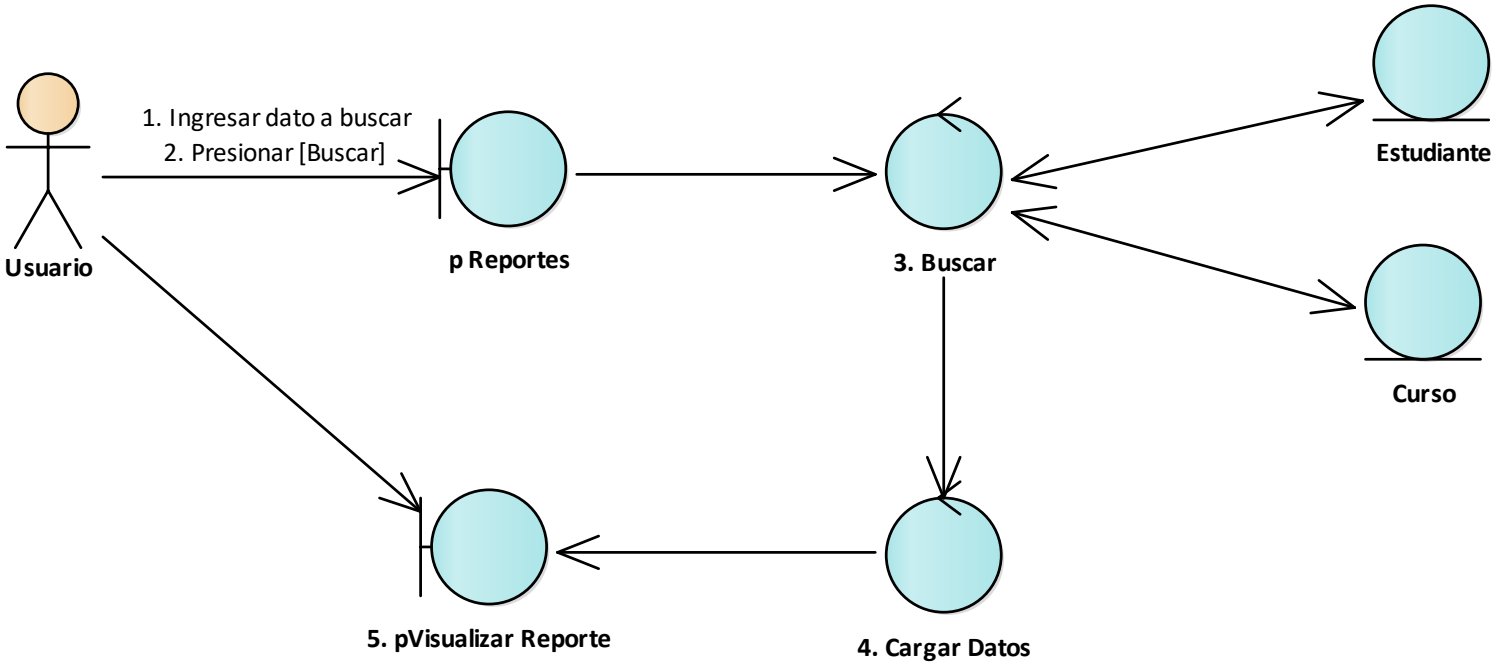


Figura 11 Diagrama de Robustez Ver Reporte

Diagramas de Secuencia

Desarrollo de aplicativo web para la gestión de cobro de mensualidades a estudiantes del preuniversitario "CENES"

Contenido:

1. Diagramas de Secuencia

1.1. Diagrama de Secuencia para Iniciar Sesión

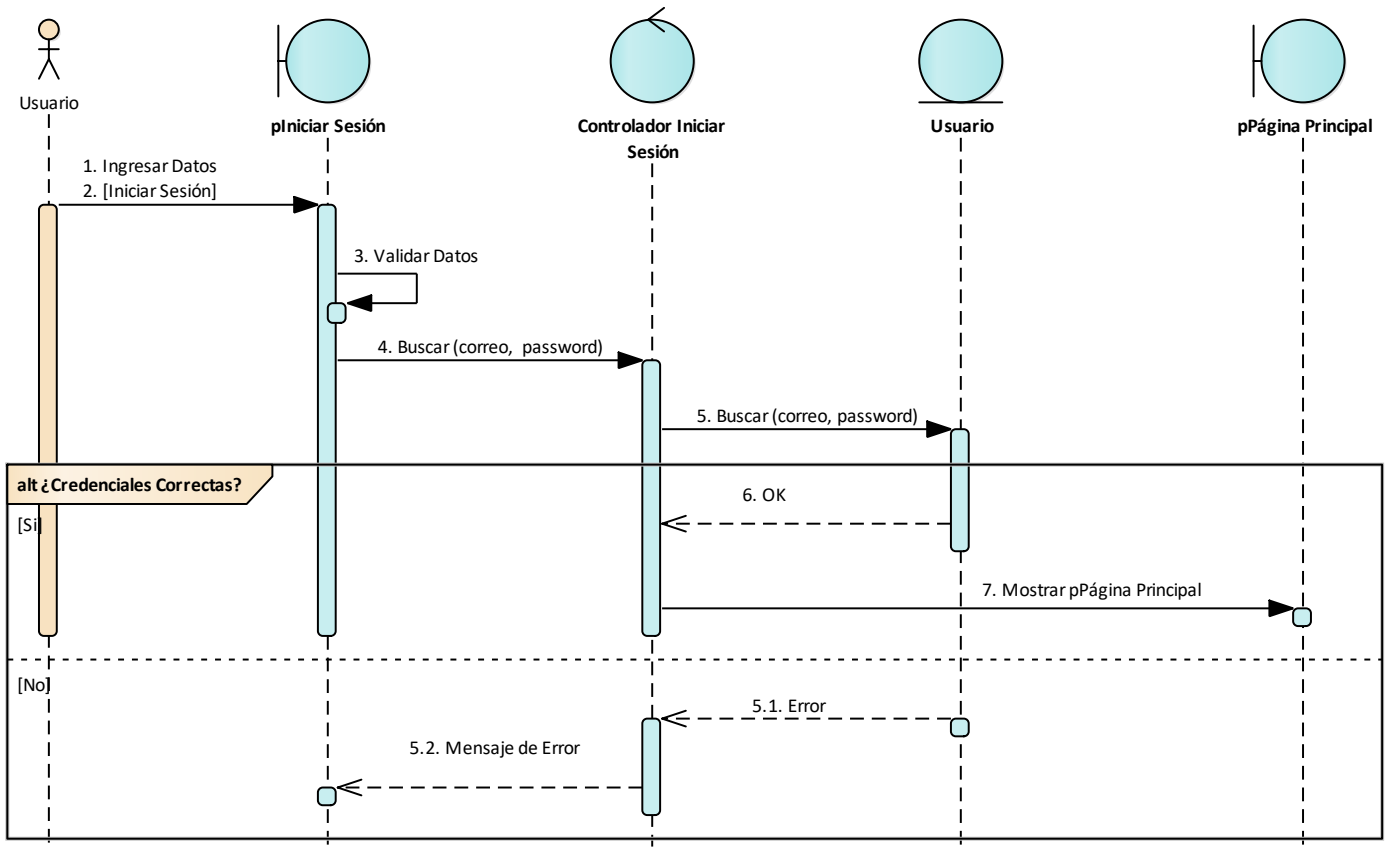


Figura 1 Diagrama de Secuencia Inicio de Sesión

1.2. Diagrama de Secuencia Crear Curso

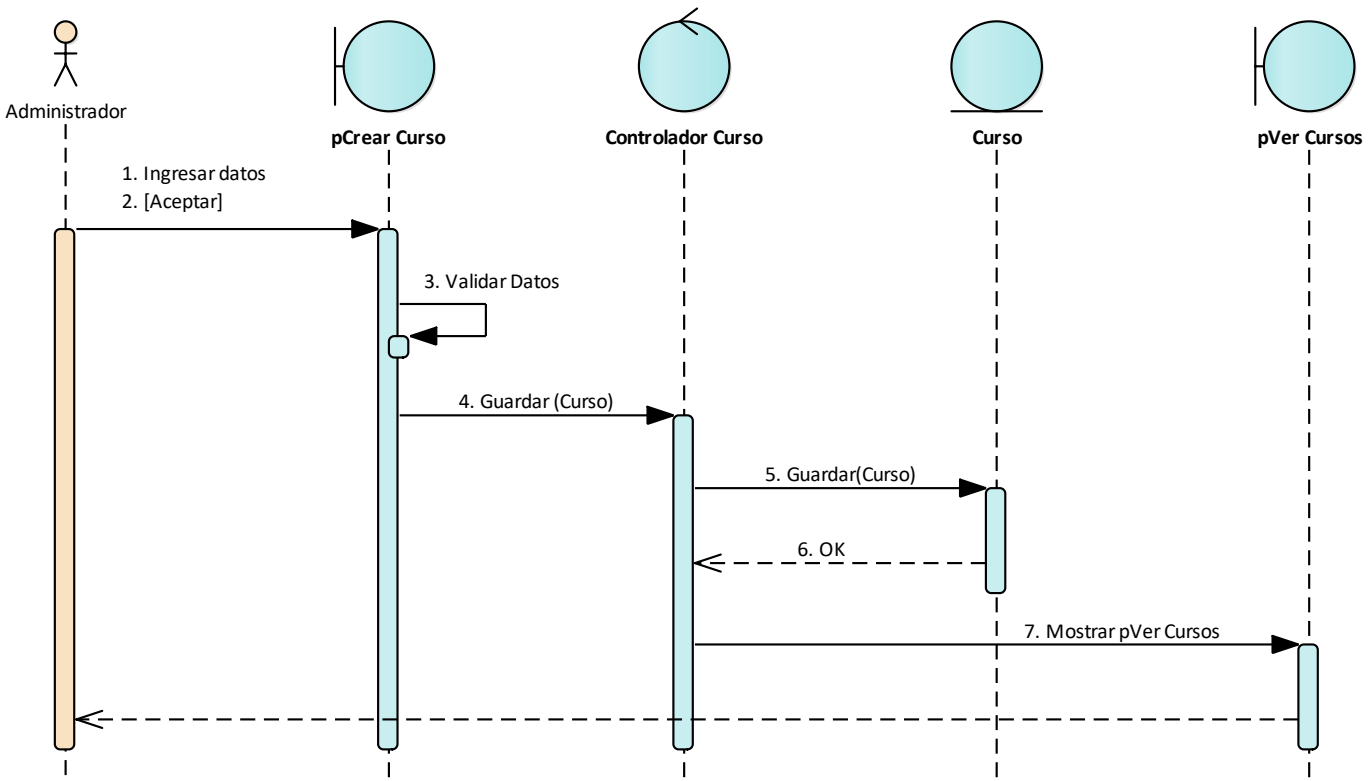


Figura 2 Diagrama de Secuencia Crear Curso

1.3. Diagrama de Secuencia Editar Curso

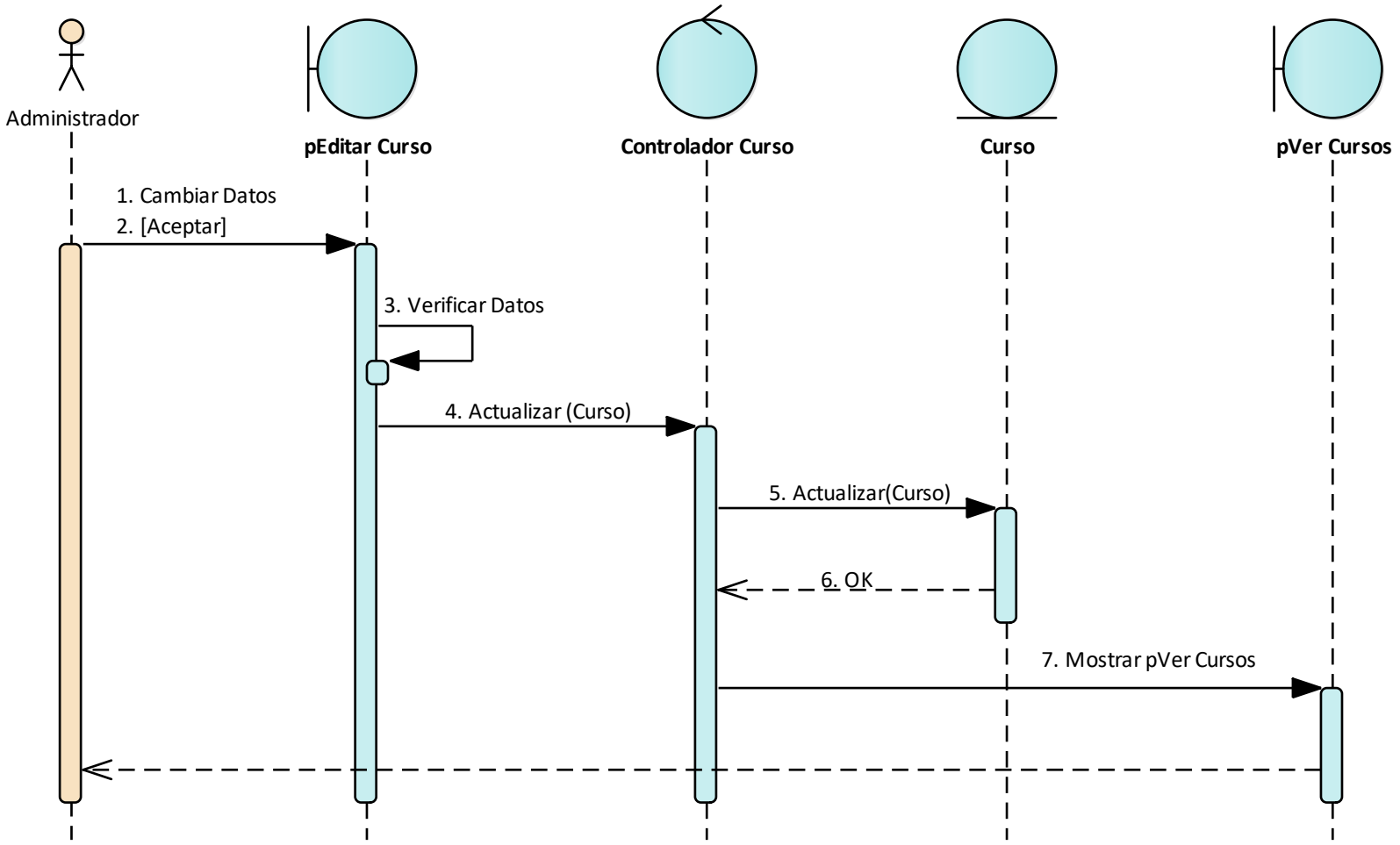


Figura 3 Diagrama de Secuencia Editar Curso

1.4. Diagrama de Secuencia Buscar Curso

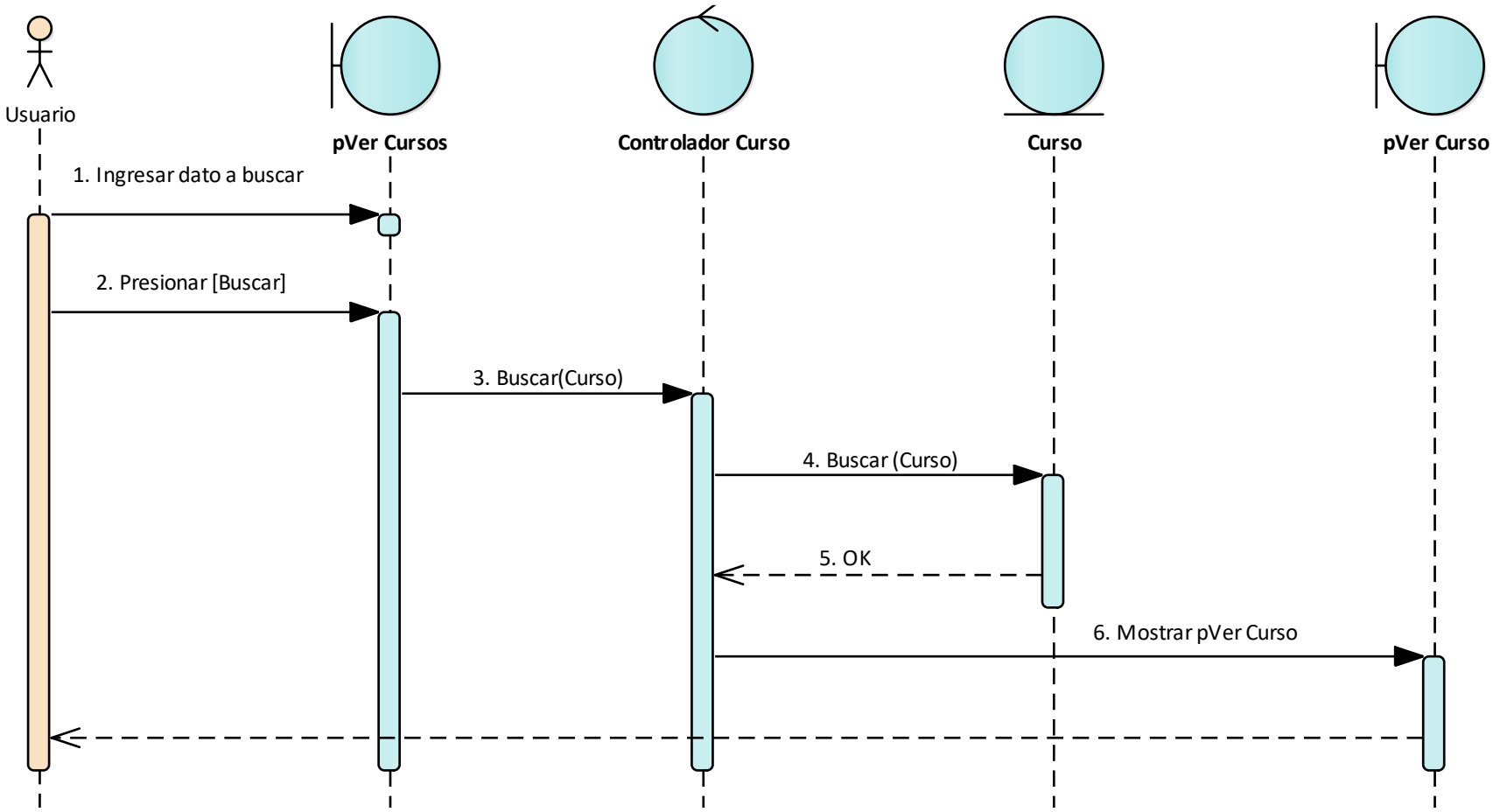


Figura 4 Diagrama de Secuencia Eliminar Curso

1.5. Diagrama de Secuencia Eliminar Curso

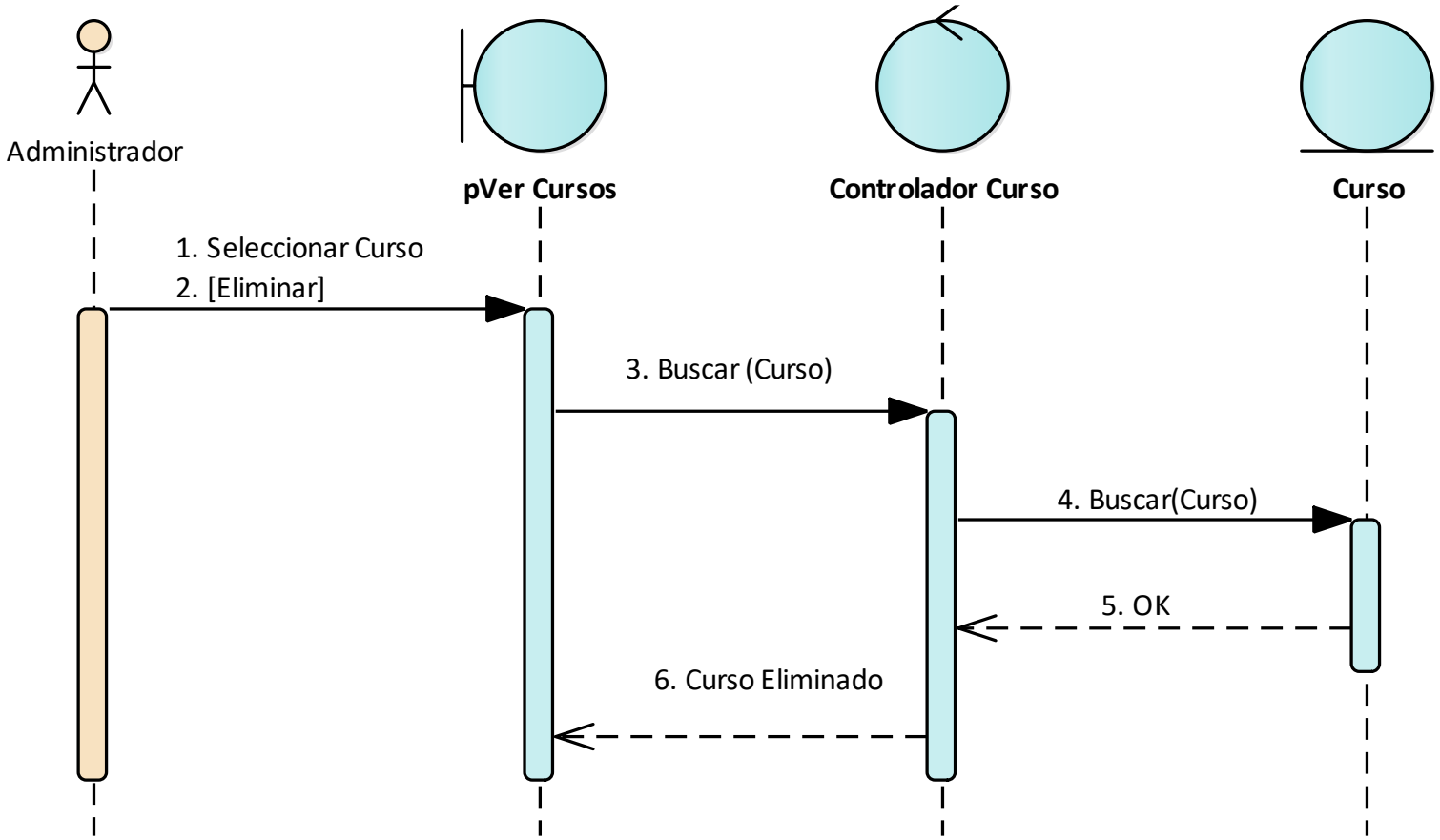


Figura 5 Diagrama de Secuencia Eliminar Curso

1.6. Diagrama de Secuencia Crear Estudiante

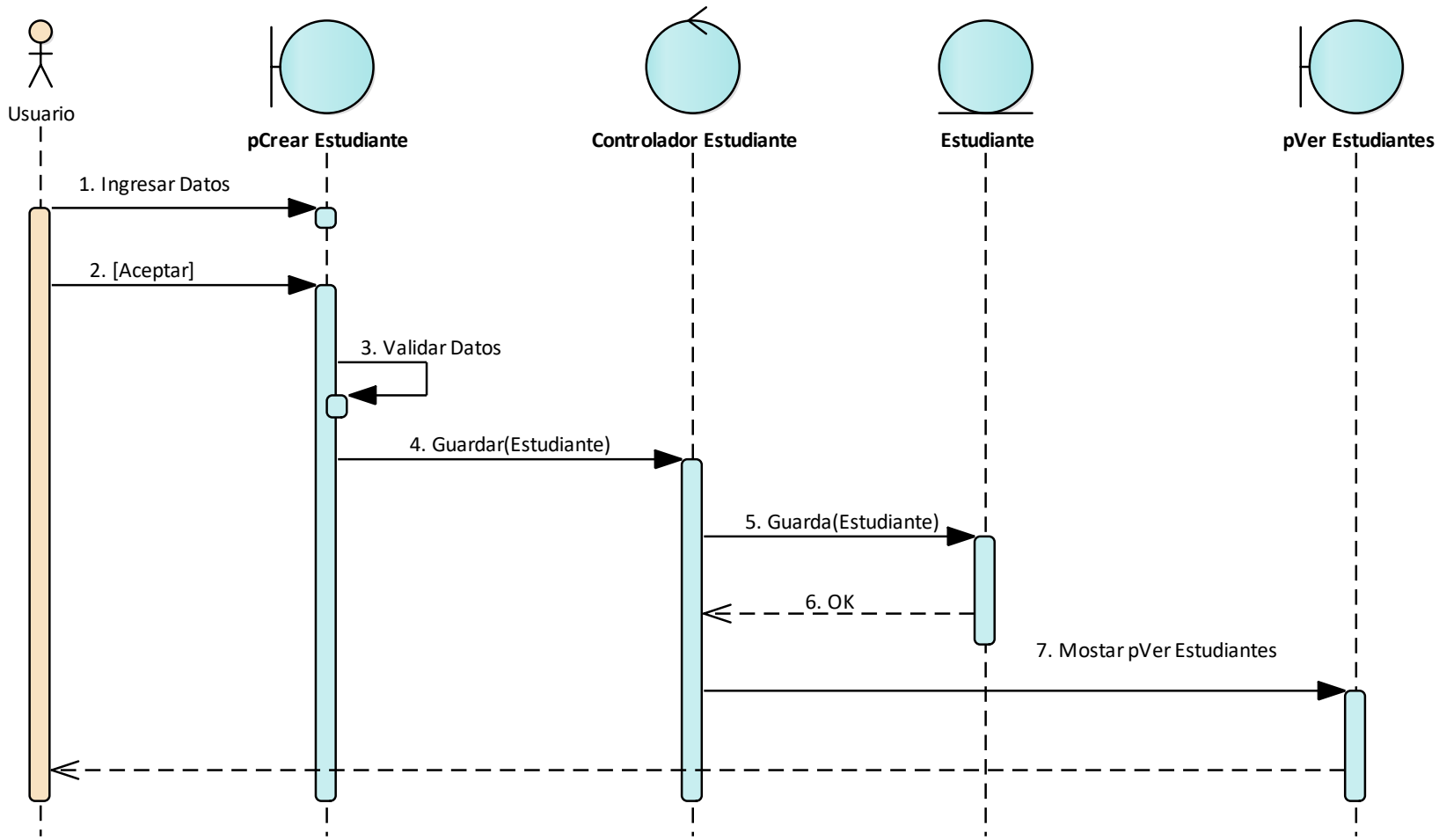


Figura 6 Diagrama de Secuencia Editar Estudiante

1.7. Diagrama de Secuencia Editar Estudiante

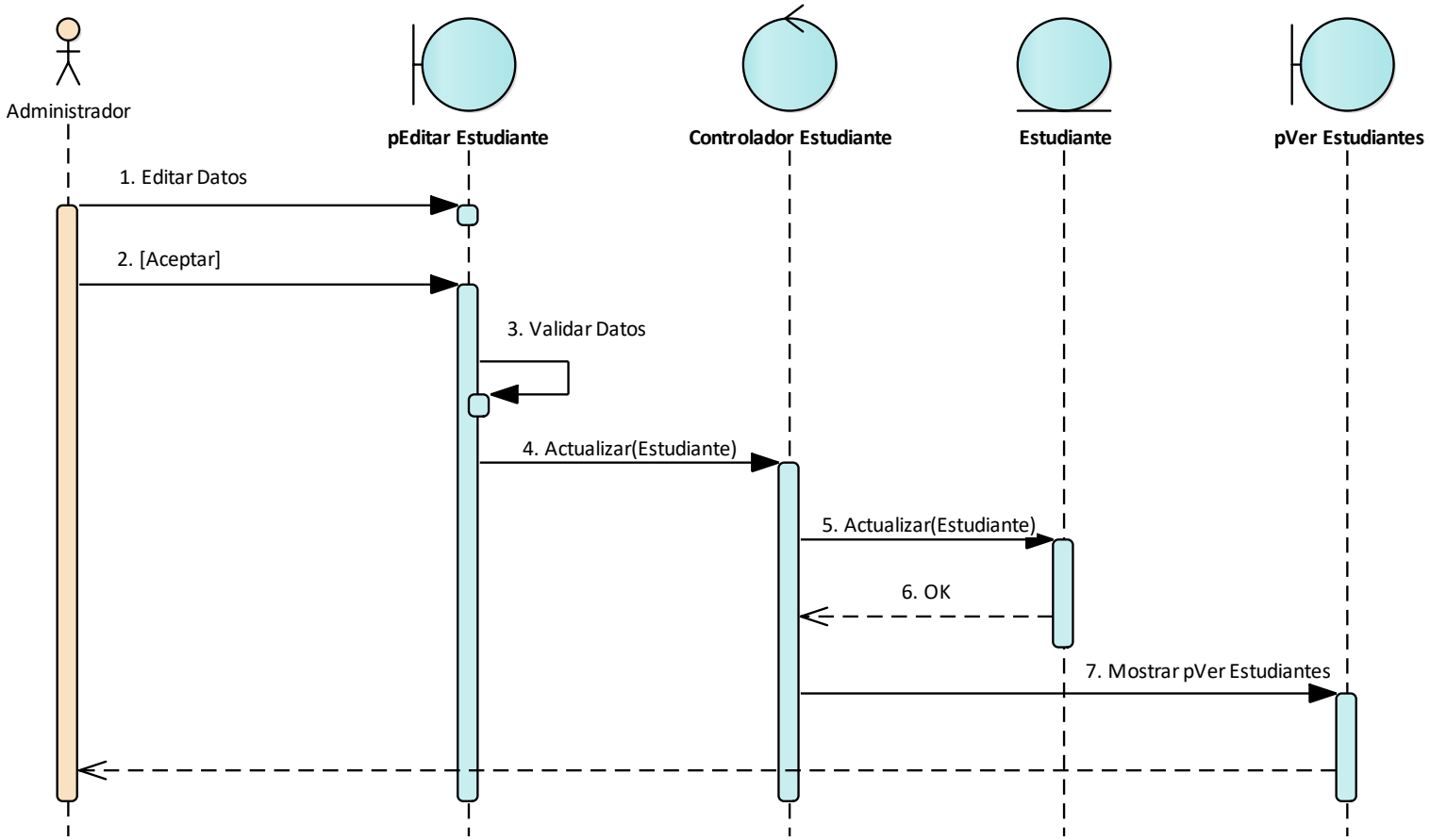


Figura 7 Diagrama de Secuencia Editar Estudiante

1.8. Diagrama de Secuencia Buscar Estudiante

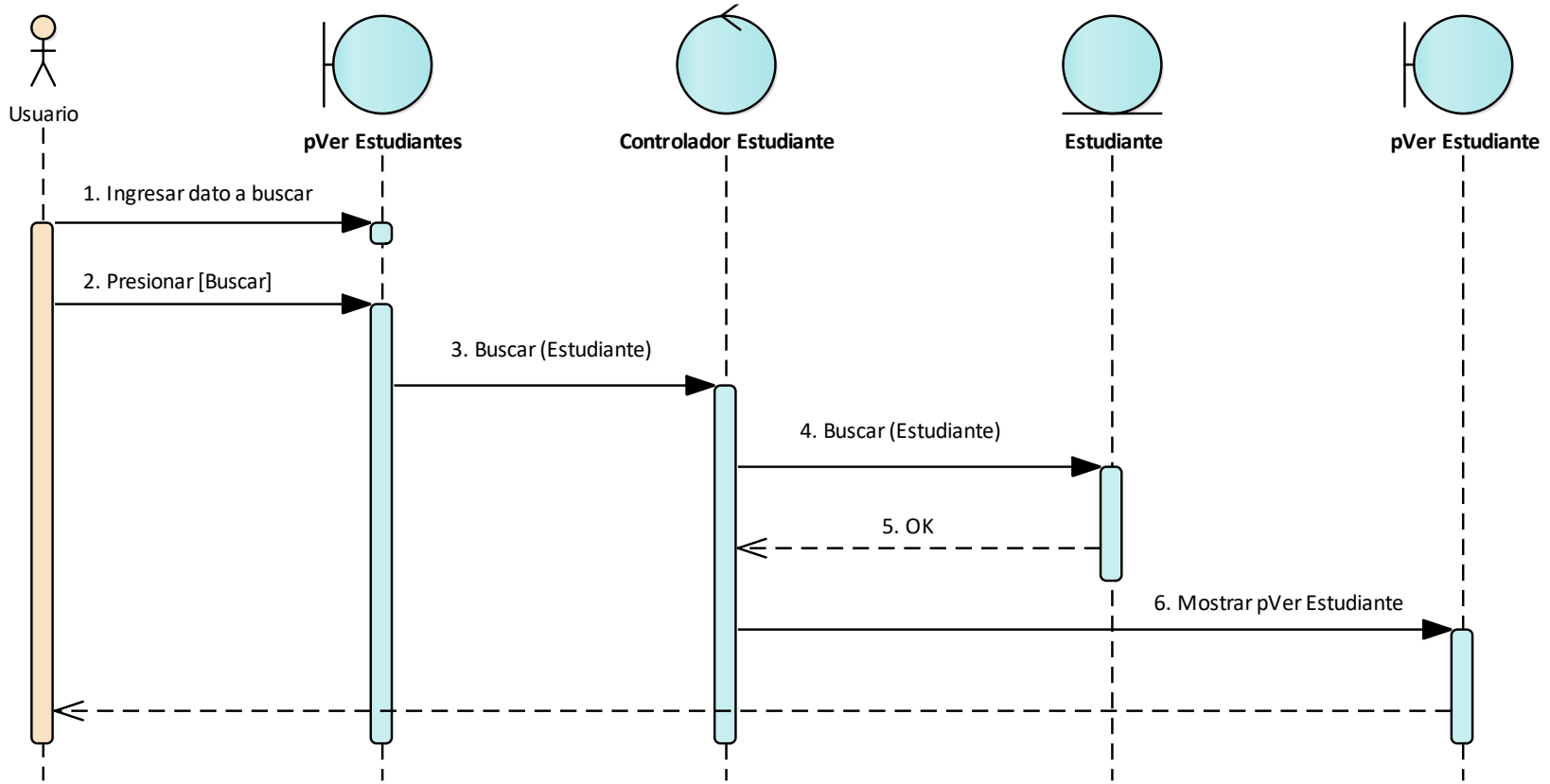


Figura 8 Diagrama de Secuencia Buscar Estudiante

1.9. Diagrama de Secuencia Registrar Pago

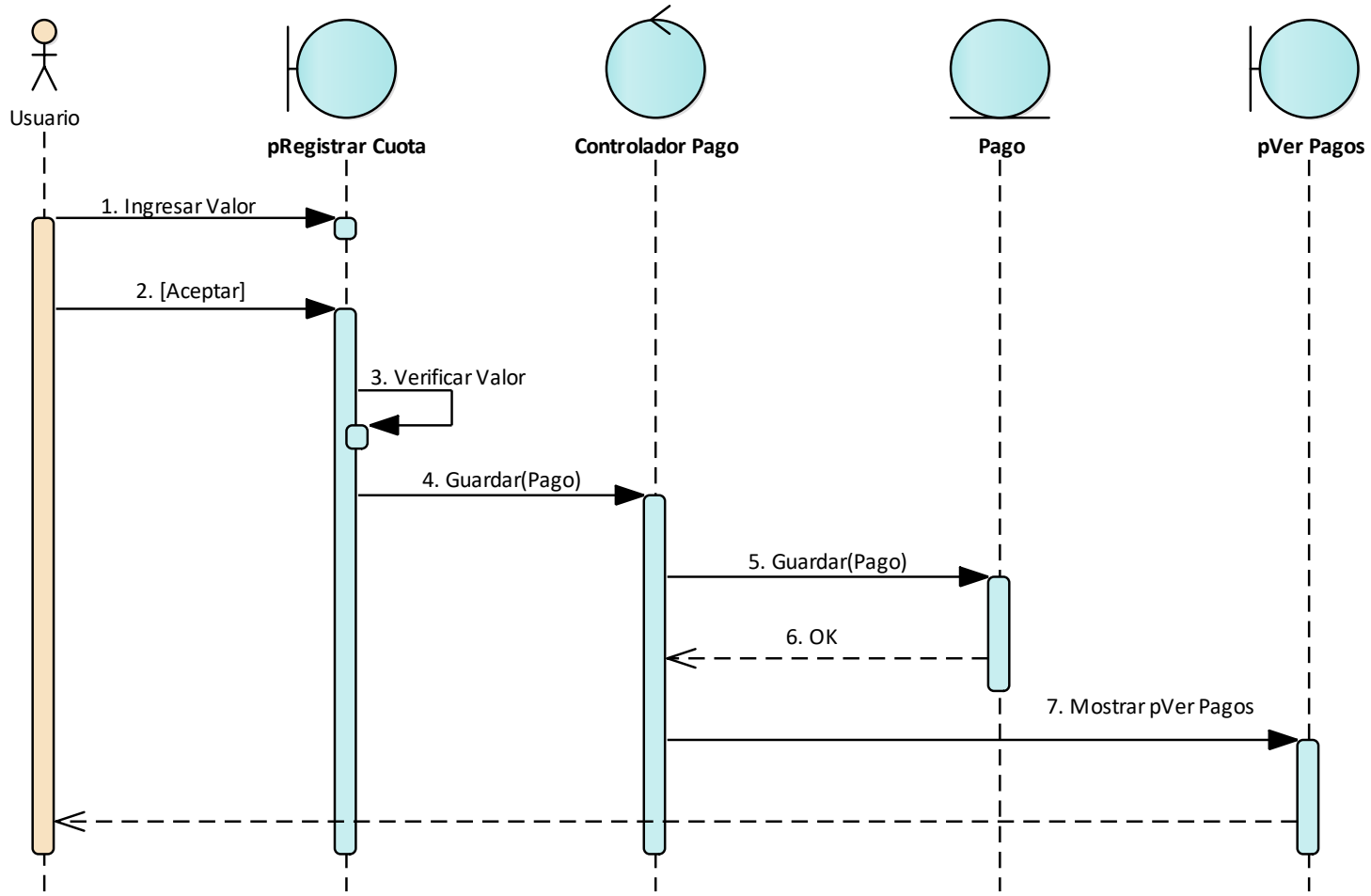


Figura 9 Diagrama de Secuencia Registrar Pago

1.10. Diagrama de Secuencia Buscar Pago

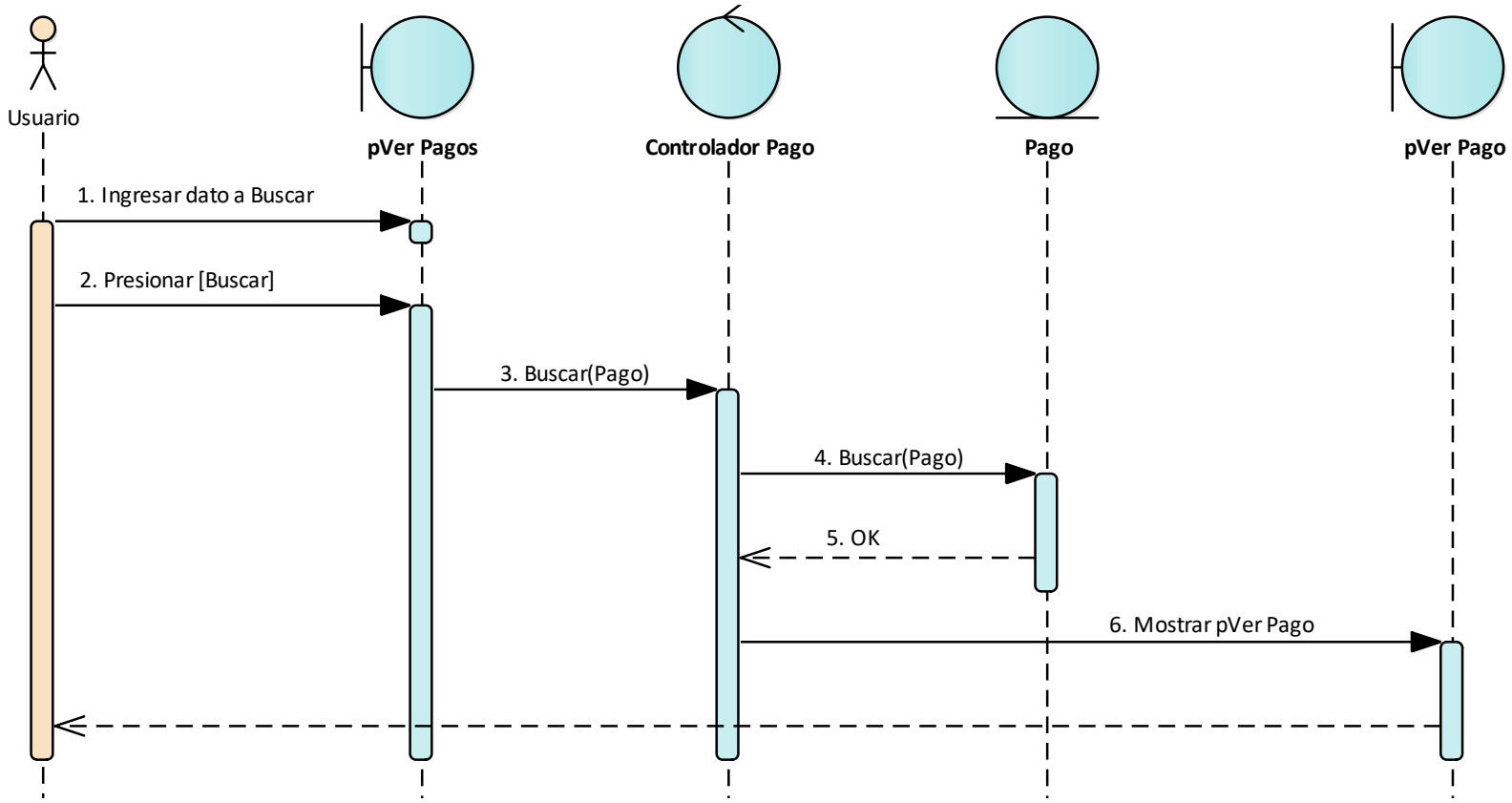


Figura 10 Diagrama de Secuencia Buscar Pago

1.11. Diagrama de Secuencia Ver Reporte

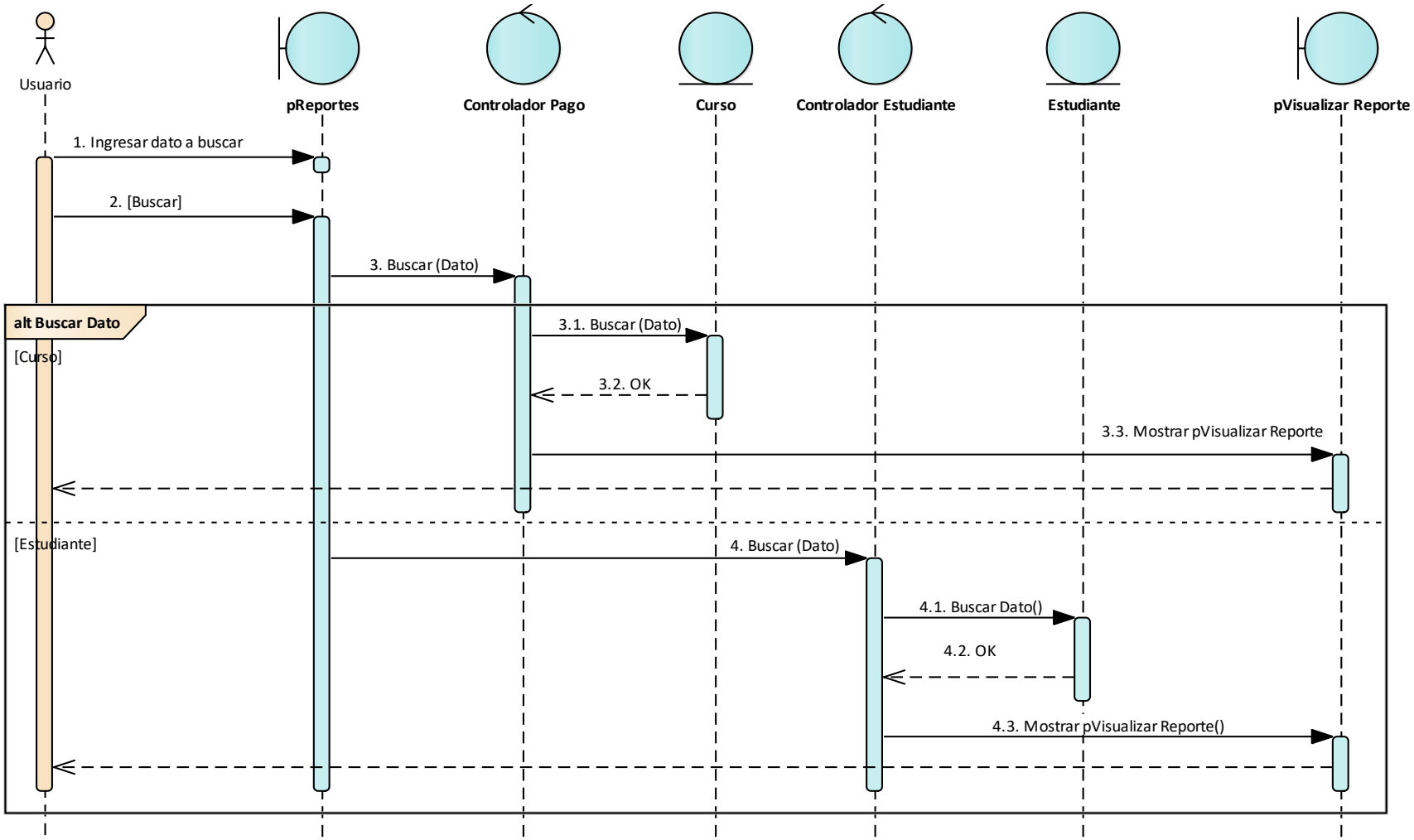


Figura 11 Diagrama de Secuencia Ver Reporte

Pruebas de Caja Negra al Servicio Web API REST

Pruebas de Caja Negra al Servicio Web API REST

Proyecto: Desarrollo de un aplicativo web para la gestión de cobros del preuniversitario “CENES”

VALIDACIÓN DE LAS PRUEBAS DE CAJA NEGRA AL SERVICIO WEB API REST PARA EL APLICATIVO WEB DE GESTIÓN DE COBROS DEL PREUNIVERSITARIO "CENES".

E = Encargado de la elaboración del documento.

R = Encargado de la revisión del documento.

A = Encargado de la aprobación del documento de pruebas al servicio web API REST.

NOMBRE	CARGO	E	R	A	FIRMA
Carla Isabel Troya Capa	Tesista	X			
Edwin René Guamán Quinche	Revisor Técnico		X	X	Firmado electrónicamente por: EDWIN RENE GU QUINCHE Firmado electrónicamente por: CARLA ISABEL TROYA CAPA

1. Introducción

El presente documento muestra las pruebas al servicio web API REST del aplicativo web para la gestión de cobros del preuniversitario “CENES”, para comprobar el correcto funcionamiento de los end points, controlando que las respuestas devueltas sean correctas.

2. Propósito

Detectar posibles errores cometidos en la implementación del servicio web API REST, así mismo, probar que los end points respondan de manera correcta las peticiones; para lo cual se utilizó la herramienta de software Postman.

3. Referencias

Referencia Sitio Web Postman https://www.postman.com/

4. Pruebas de Caja Negra al Servicio Web API REST

Para las pruebas del API REST se utilizó la herramienta Postman, la cual permitió el envío de peticiones http (get, put, delete, post) hacia el servicio web API REST, obteniendo una respuesta en formato json

Tabla 1 Petición para crear un curso

URL	Método	Tiempo de ejecución	Descripción
http://127.0.0.1:8000/api/courses/	POST	32 ms	Permite Crear un Curso
RESULTADO			

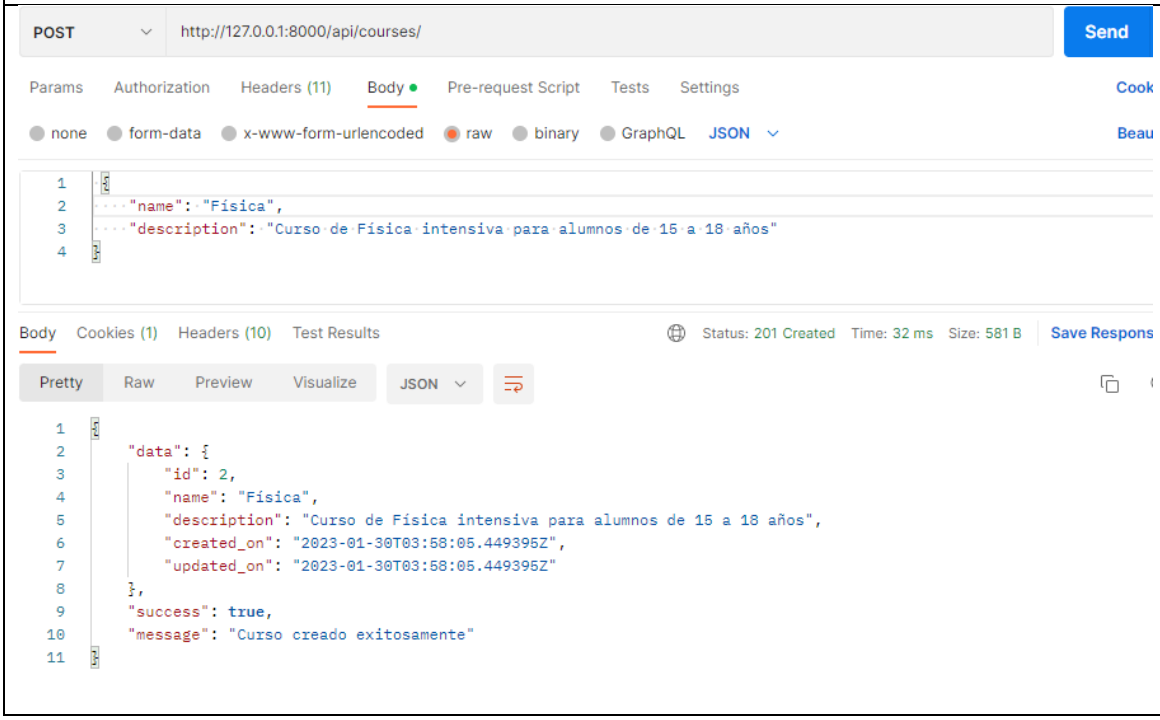


Tabla 2 Petición para editar un curso

URL	Método	Tiempo de ejecución	Descripción
http://127.0.0.1:8000/api/courses/1/	PUT	17 ms	Permita la edición de un curso
RESULTADO			

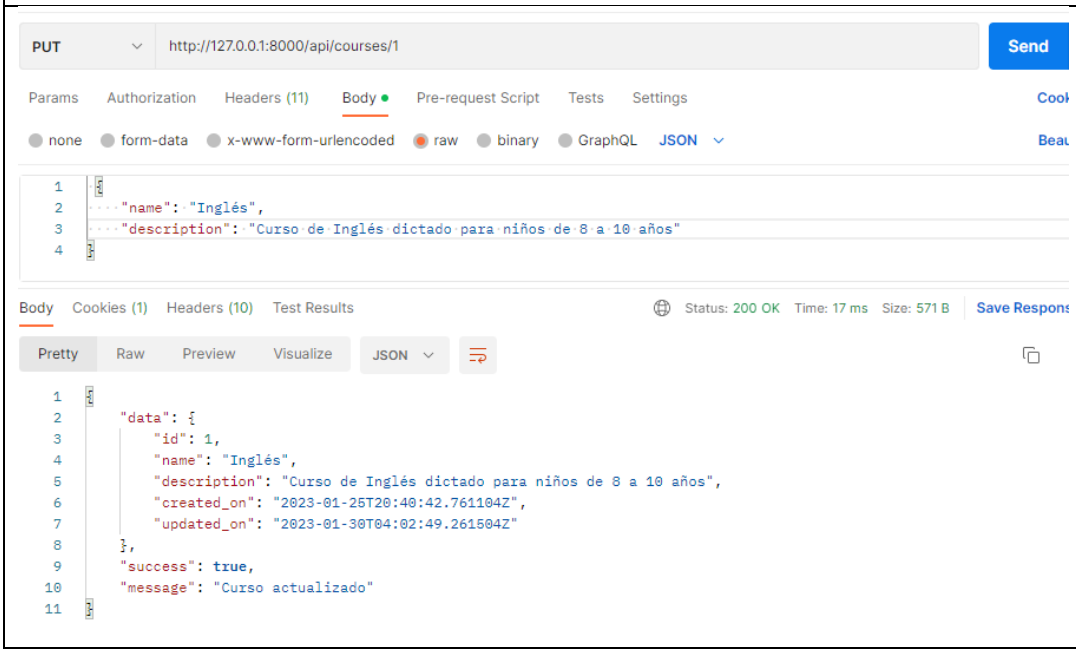


Tabla 3 Petición para eliminar un curso

URL	Método	Tiempo de ejecución	Descripción
http://127.0.0.1:8000/api/courses/1/	DELETE	26 ms	Permite eliminar un curso
RESULTADO			

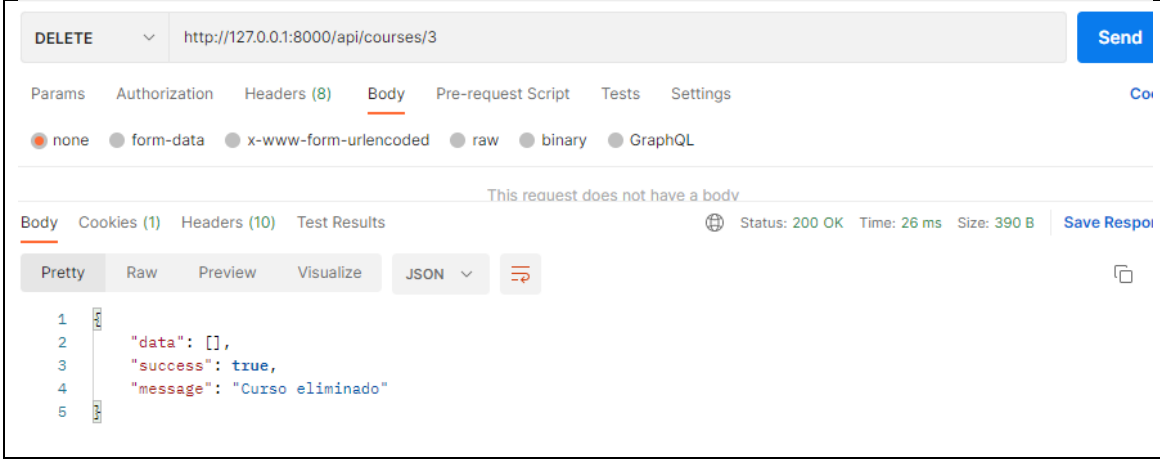


Tabla 4 Petición para crear una cohorte

URL	Método	Tiempo de ejecución	Descripción
http://127.0.0.1:8000/api/cohortes/	POST	27 ms	Permite la creación de una cohorte
RESULTADO			

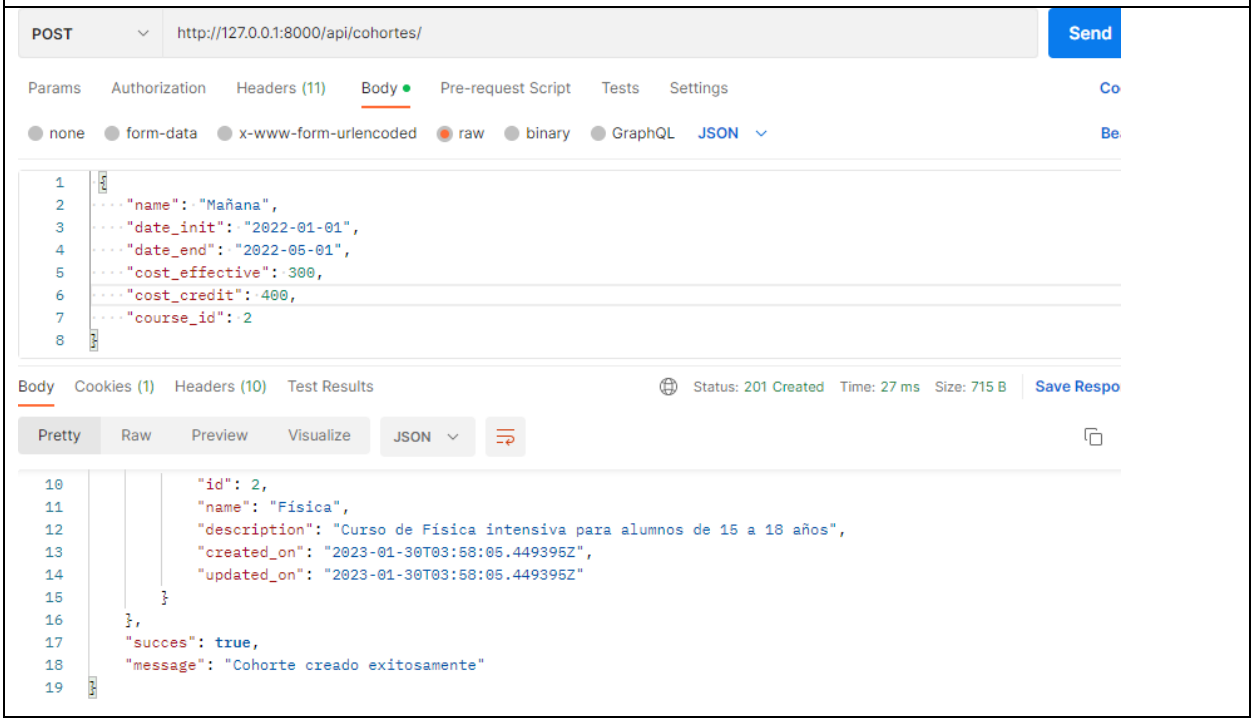


Tabla 5 Petición para editar una cohorte

URL	Método	Tiempo de ejecución	Descripción
http://127.0.0.1:8000/api/cohortes/1	PUT	37 ms	Permite la edición de una cohorte
RESULTADO			

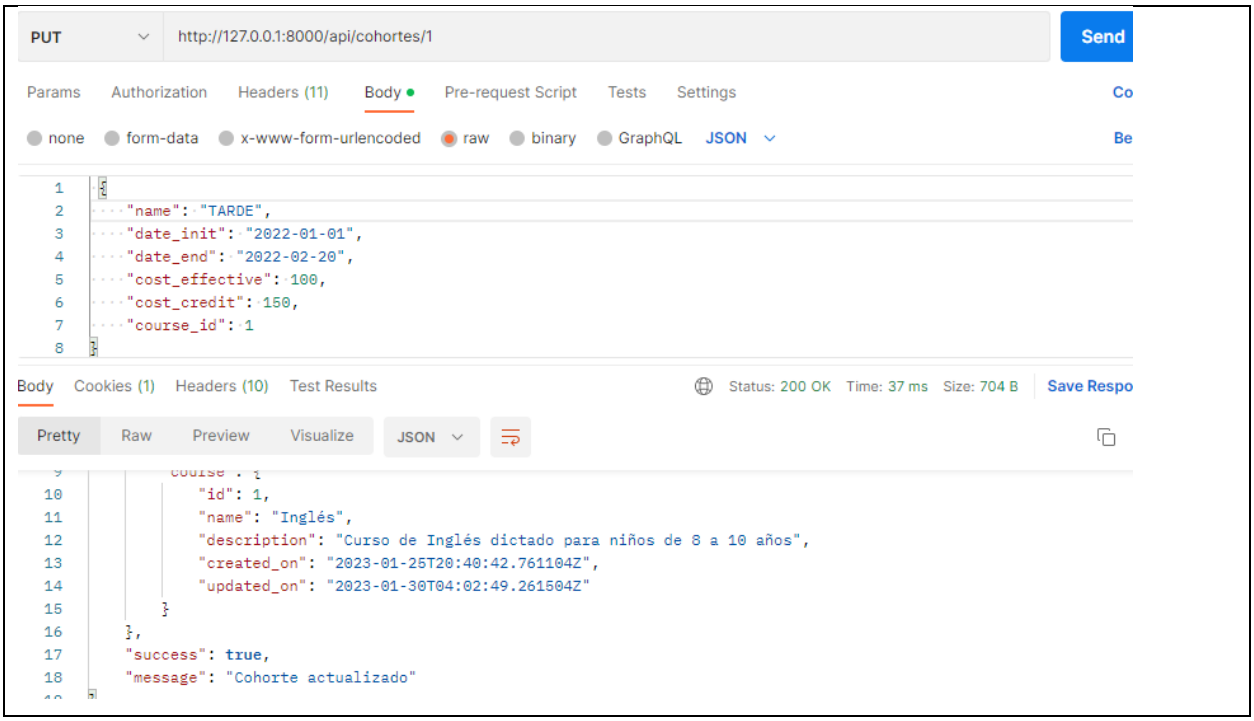


Tabla 6 Petición para crear un estudiante

URL	Método	Tiempo de ejecución	Descripción
http://127.0.0.1:8000/api/students/	POST	52 ms	Permite crear un estudiante
RESULTADO			

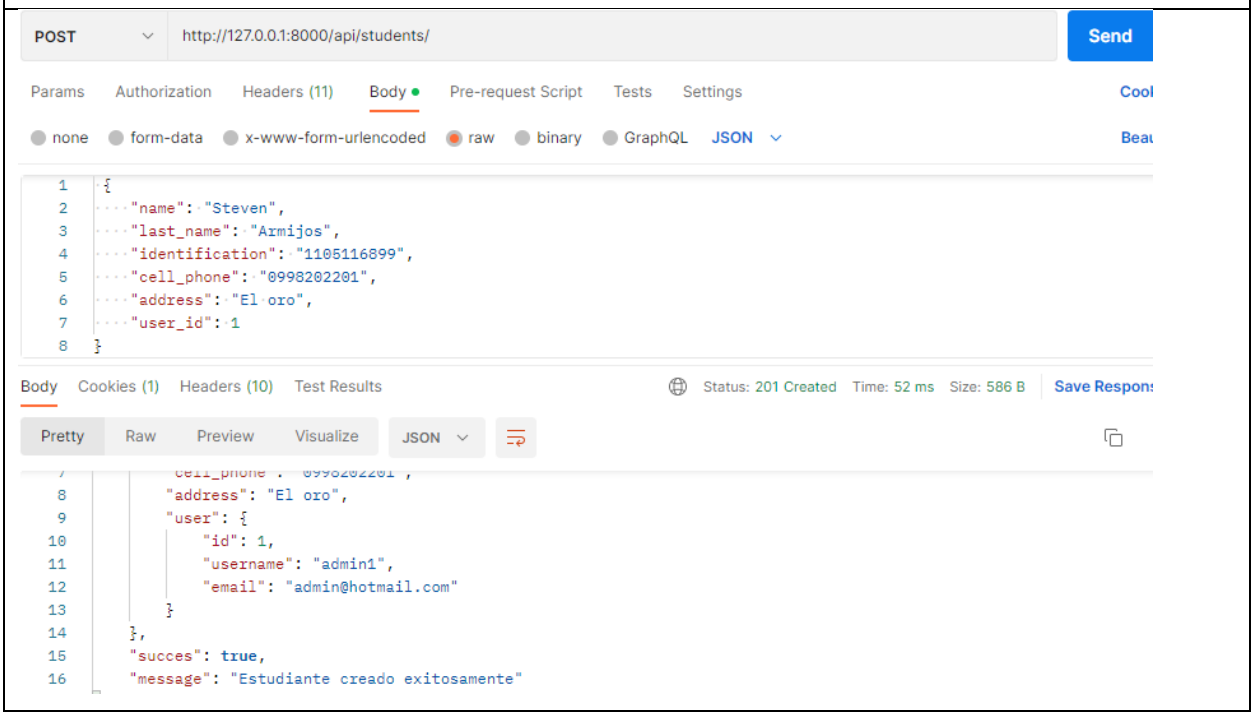


Tabla 7 Petición para editar un estudiante

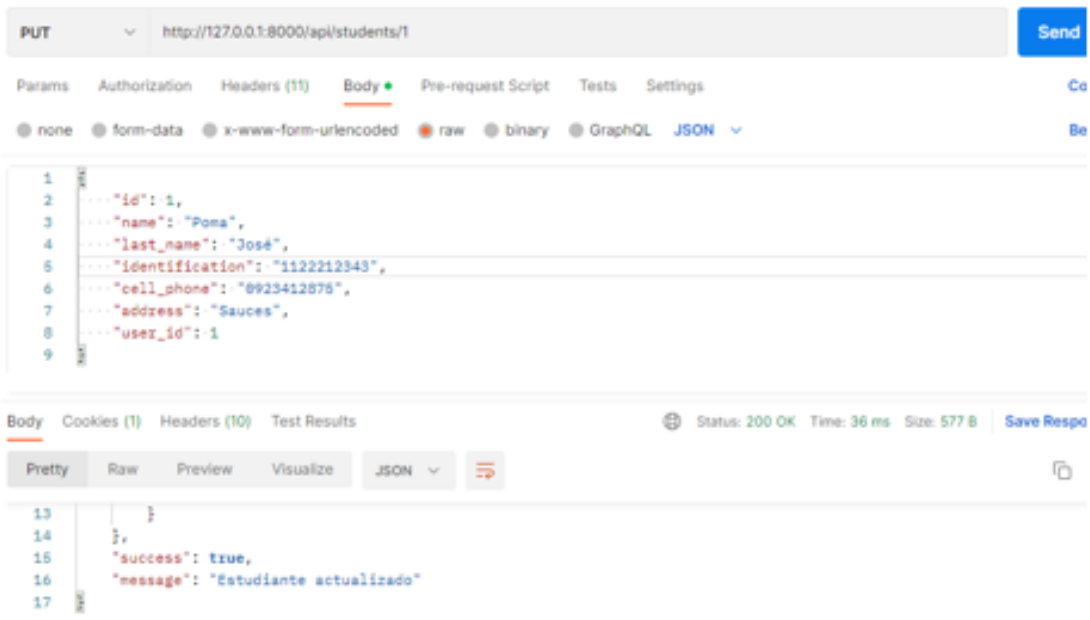
URL	Método	Tiempo de ejecución	Descripción
http://127.0.0.1:8000/api/students/	PUT	36 ms	Permite la edición de un estudiante
RESULTADO			
			

Tabla 8 petición para crear una matrícula

URL	Método	Tiempo de ejecución	Descripción
http://127.0.0.1:8000/api/enrollements/	POST	36 ms	Permite la creación de una matrícula
RESULTADO			

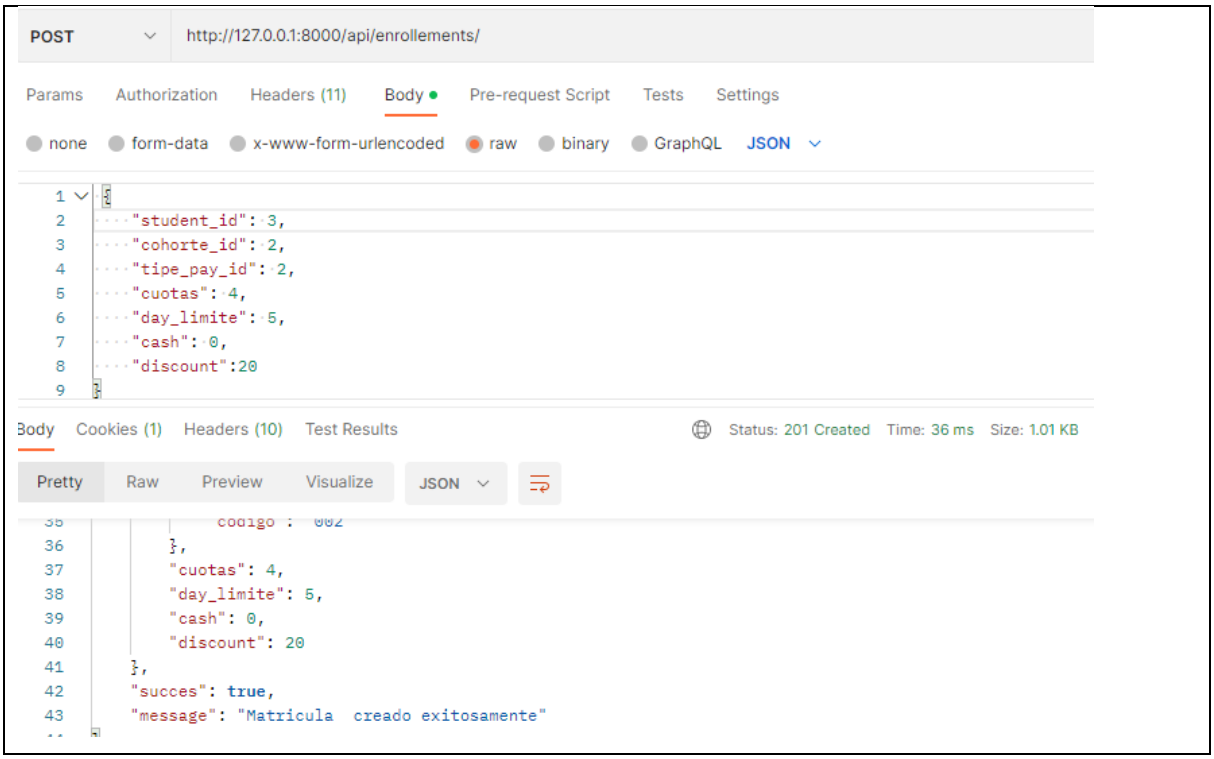


Tabla 9 Petición para registrar un pago

URL	Método	Tiempo de ejecución	Descripción
http://127.0.0.1:8000/api/payments/	POST	32 ms	Permite registrar un pago en base a una matrícula creada
RESULTADO			

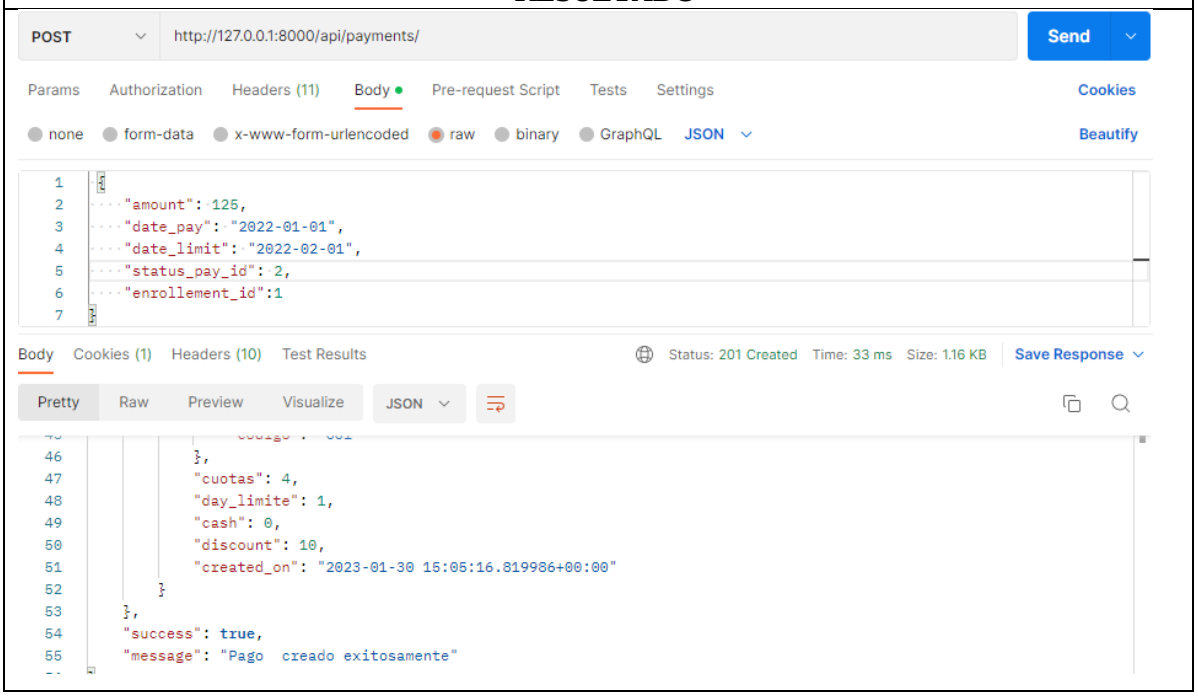


Tabla 10 Petición para editar un pago

URL	Método	Tiempo de ejecución	Descripción
http://127.0.0.1:8000/api/payments/	PUT	37 ms	Permite la edición de un pago
RESULTADO			

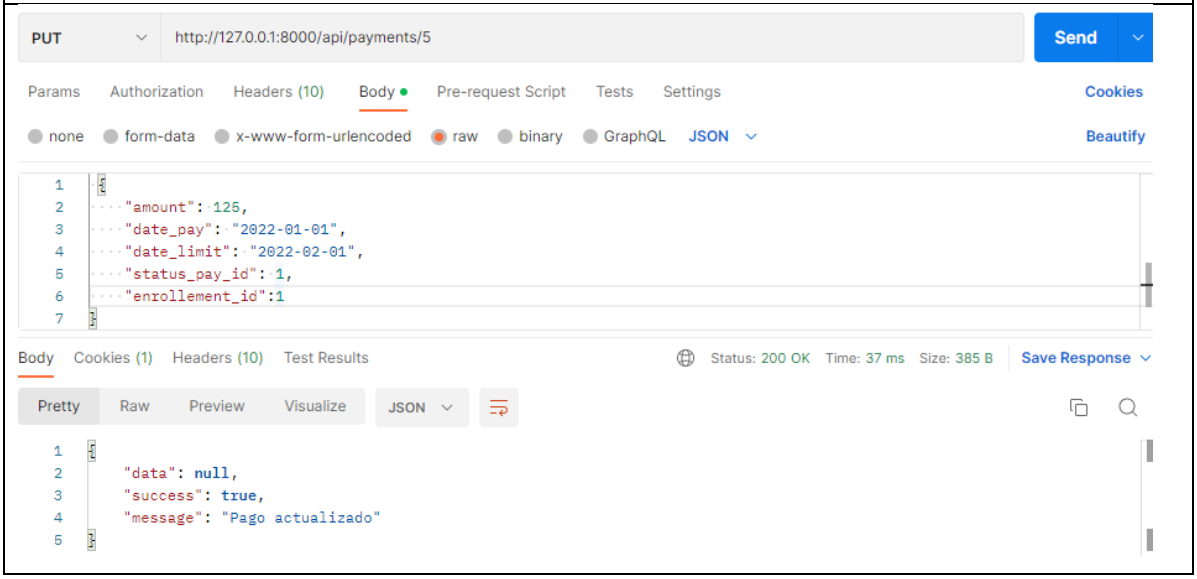


Tabla 11 Petición para el inicio de sesión

URL	Método	Tiempo de ejecución	Descripción
http://127.0.0.1:8000/auth/login/	POST	326 ms	Permite el inicio de sesión de un usuario
RESULTADO			

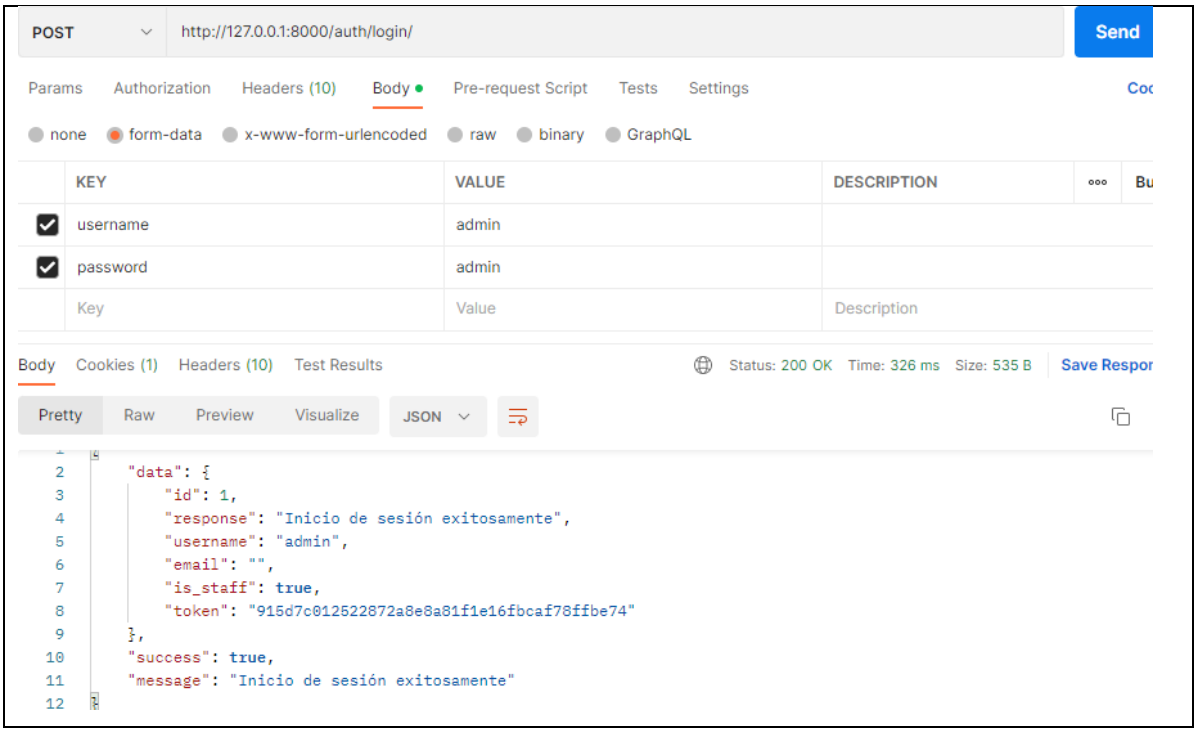


Tabla 12 Petición para crear un usuario

URL	Método	Tiempo de ejecución	Descripción
http://127.0.0.1:8000/auth/register/	POST	385 ms	Permite crear un usuario
RESULTADO			

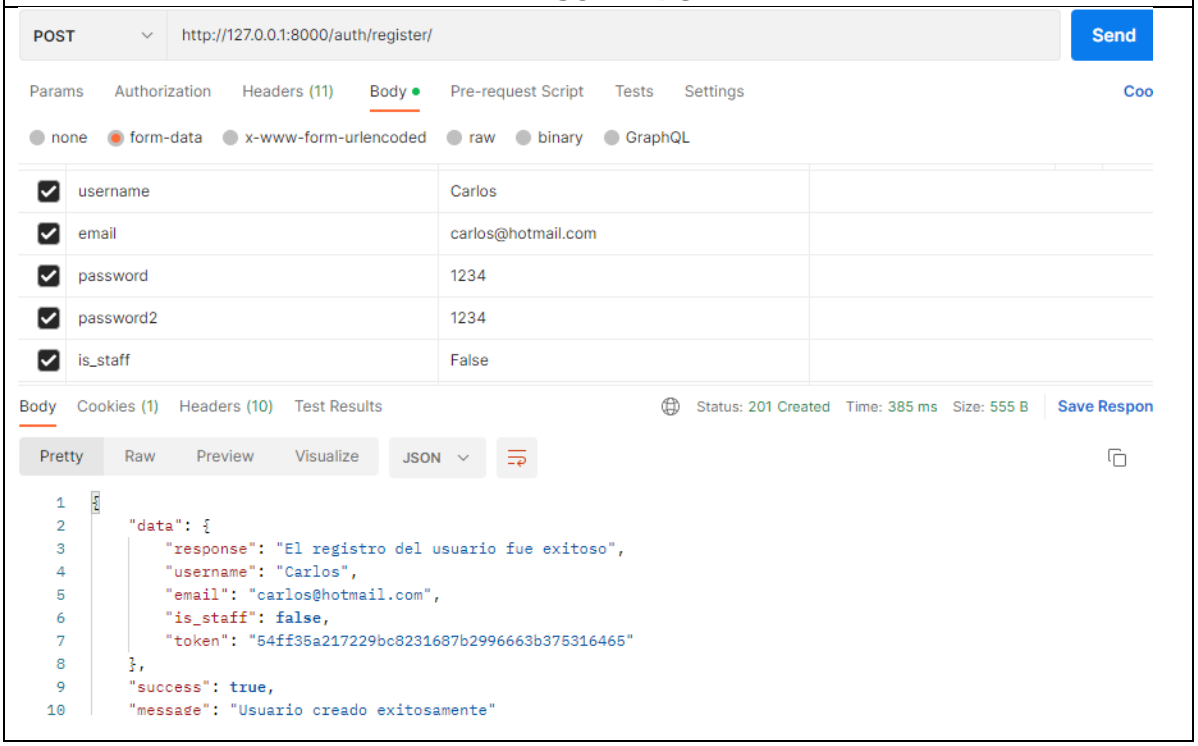


Tabla 13 Petición para editar un usuario

URL	Método	Tiempo de ejecución	Descripción
http://127.0.0.1:8000/auth/users/2	PUT	19 ms	Permite la edición de un usuario creado
RESULTADO			

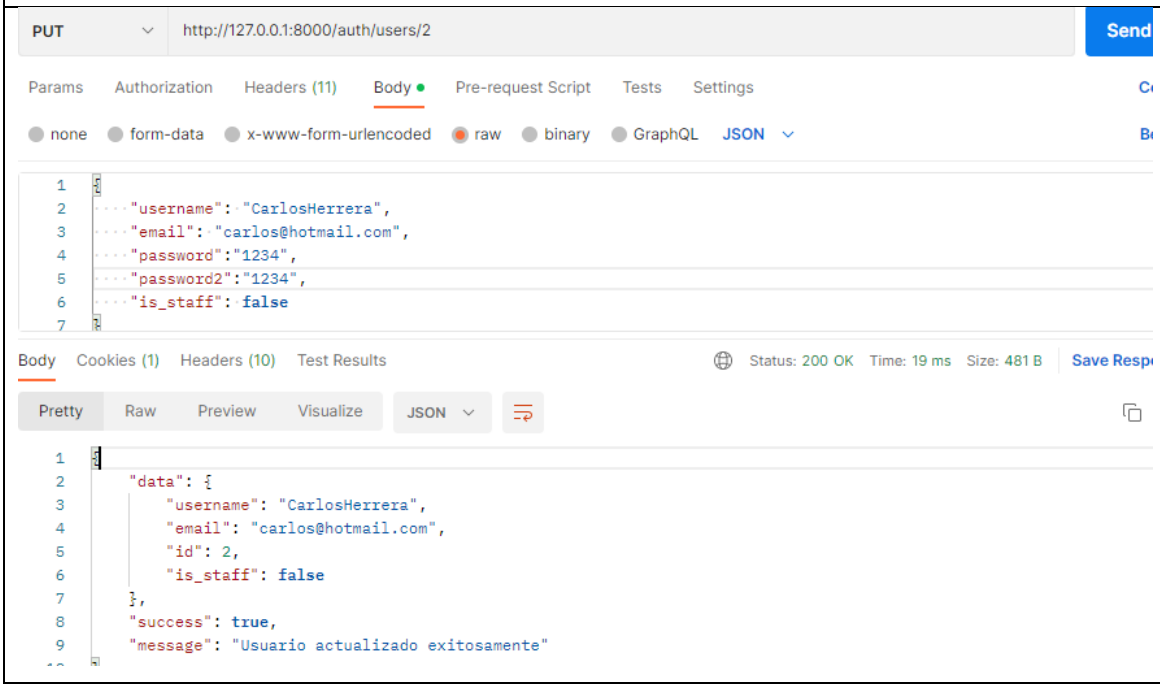
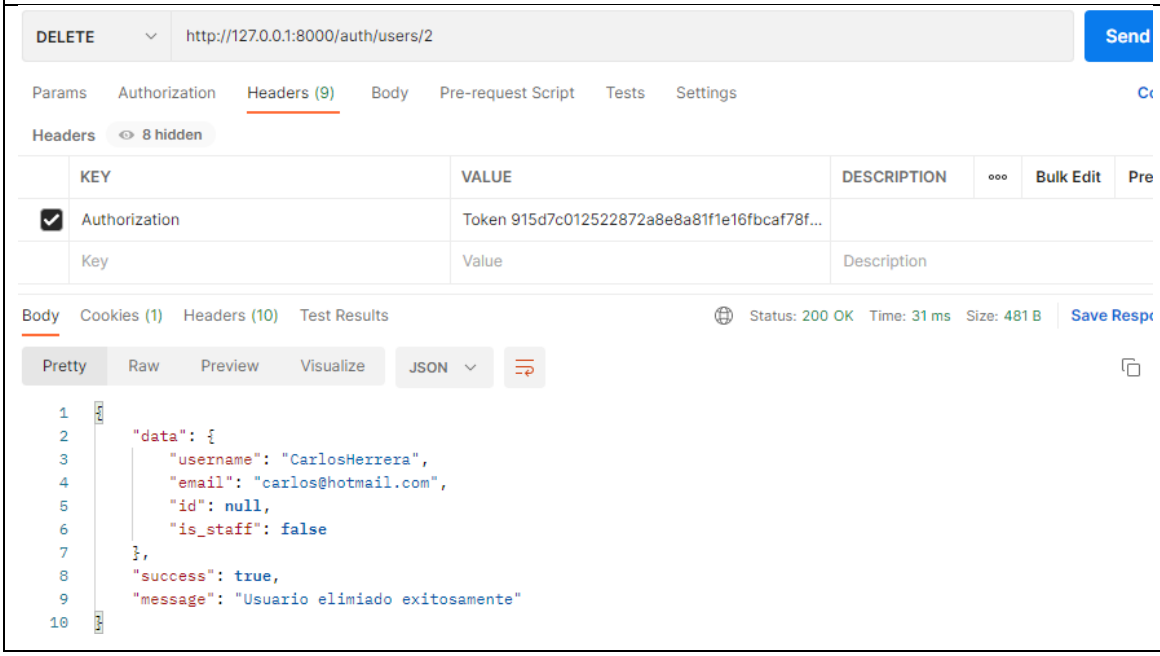


Tabla 14 Petición para eliminar un usuario

URL	Método	Tiempo de ejecución	Descripción
http://127.0.0.1:8000/auth/users/2	DELETE	31 ms	Permite eliminar un usuario
RESULTADO			



5. Lista de Requisitos evaluados.

Realizadas las pruebas se tomó a consideración realizar una tabla que se muestra a continuación, para realizar un chequeo para verificar que las pruebas de caja negra cumplen con los requisitos del software.

Tabla 15 Chequeo para las Pruebas de Caja Negra

Casos de Prueba de Caja Negra			
Criterio	**CUMPLE**		**Observación**
	Si	**No**	
Iniciar sesión y mostrar mensaje de éxito o motivo del error.	X		
Crear un curso y mostrar mensaje de éxito o motivo del error	X		
Editar un curso y mostrar mensaje de éxito o motivo del error	X		
Eliminar un curso y mostrar mensaje de éxito o motivo del error	X		
Buscar un curso	X		
Crear una cohorte, a partir de un curso y mostrar mensaje de éxito o motivo del error.		X	Error al crear una cohorte
Editar una cohorte y mostrar mensaje de éxito o motivo del error.	X		
Eliminar una cohorte, siempre y cuando no haya estudiantes matriculados, mostrar mensaje de éxito o motivo del error.	X		
Buscar una Cohorte	X		
Crear un Estudiante y mostrar mensaje de éxito o motivo del error.	X		
Editar un Estudiante y mostrar mensaje de éxito o motivo del error.	X		
Buscar un Estudiante.	X		
Crear una matrícula y mostrar mensaje de éxito o motivo del error.	X		
Eliminar matrícula y mostrar mensaje de éxito o motivo del error.		X	Error al eliminar matricula ya que está relacionada con los pagos
Registrar un pago y mostrar mensaje de éxito o motivo del error.	X		
Editar un pago y mostrar mensaje de éxito o motivo del error.	X		

6. Errores y Soluciones

Al realizar las pruebas de caja negra se encontró los siguientes errores y se detalla la solución brindada:

Tabla 16 Errores y Soluciones

ERRORES	SOLUCIÓN
Error al eliminar una matrícula, este error se dio debido a que en el modelo de enrollment, la llave ForeignKey de Payment se encontraba en RESTRICTED, lo que no permitía que se elimine una matrícula.	Al identificar el error se procedió a realizar el cambio en cuanto al valor de llave ForeignKey de Payment contenida en el modelo enrollment, el nuevo valor del ForeignKey es CASCADE, lo que permite que se elimine la matricula conjuntamente con los pagos, que es lo requerido por el cliente.
Se presento un error al momento de crear una cohorte, dado que la cohorte se crea a partir de un curso.	Al revisar el código se encontró que en el modelo Cohorte no existía la ForeignKey del modelo Course.

Anexo 7. Pruebas de Caja Blanca

Pruebas de Caja Blanca (unitarias)

Las pruebas unitarias se realizaron a cada un de los métodos con el fin de comprobar que cada uno funciona según lo previsto, por lo que los errores no se propagan en toda la aplicación. A continuación, se muestra los test aplicados a la codificación

1. **Prueba unitaria para iniciar sesión**

```
class TestSetUp(APITestCase):

    def setUp(self):
        self.login_url = '/auth/login/'
        self.user = User.objects.create_superuser(
            username='lola',
            email= faker.email(),
            password='lola'
        )

        response = self.client.post(
            self.login_url,
            {
                'username': 'lola',
                'password': 'lola'

            },
            format = 'json'
        )

        self.assertEqual(response.status_code, status.HTTP_200_OK)
        return super().setUp()
```

Figura 1 Test Case Iniciar Sesión

2. **Prueba Unitaria para registrar y buscar un curso**

En la figura 3 se muestra el código fuente para poder comprobar el funcionamiento del método. Para realizar el test se creo un archivo de ayuda que contienen datos para poder

realizar las pruebas, el mismo que se puede visualizar en la figura 2.

```
class CoursetStructure:

    def build_course_JSON(self):
        return{
            'name': "Inglés",
            'description':"Curso dictado para jovenes de 11 a 15 años"
        }

    def create_course(self):
        return Course.objects.create(**self.build_course_JSON())
```

Figura 2 Datos para Test Case Course

```
class CourseTestCase(TestSetUp):
    url= '/api/course/'

    def test_search_course(self):
        course =CourseStructure().create_course()
        response  = self.client.get(
            self.url + '<int:pk>',
        {

        },
        format= 'json'
        )

        self.assertEqual(response.status_code, status.HTTP_200_OK)
        self.assertEqual(response.data['id'], course )

    def test_search_course_error(self):
        course = CourseStructure().create_course()
        response = self.client.get(
            self.url + '<int:pk>',
            {
            'id': 1
            },
            format ='json'

        )
        self.assertEqual(response.status_code, status.HTTP_400_BAD_REQUEST)
        self.assertEqual(course.id,'1')
        self.assertEqual(response.data['mesagge'], 'No se ha encontrado el
curso')

    def test_create_course(self):
        course = CourseStructure().build_course_JSON()
        response = self.client.post(
            self.url + '',
            course,
            format='json'
        )

        self.assertEqual(response.status_code, status.HTTP_201_CREATED)
        self.assertDictEqual(Course.objects.all().count(), 1)
        self.assertEqual(response.data.course['id'], course.id)
```

Figura 3 Test Case de Course

3. Prueba Unitaria para registrar y buscar una cohorte

```python
class CohorteStructure:

    def build_cohorte_JSON(self):
        return {
            "name": "Mañana",
            "date_init": "2022-01-01",
            "date_end": "2022-05-01",
            "cost_effective": 300,
            "cost_credit": 400,
            "course_id": 1
        }

    def create_cohorte(self):
        return Cohorte.objects.create(**self.build_cohorte_JSON())
```

Figura 4 Test Case de Cohorte

```python
class CohorteTestCase(TestSetUp):
    url= '/api/cohorte/'

    def test_search_cohorte(self):
        cohorte = CohorteStructure().create cohorte()
        response  = self.client.get(
            self.url + '<int:pk>',
        {
        },
        format= 'json'
        )
        self.assertEqual(response.status code, status.HTTP 200 OK)
        self.assertEqual(response.data['id'], cohorte )
    def test_search_cohorte_error(self):
        cohorte = CohorteStructure().create cohorte()
        response = self.client.get(
            self.url + '<int:pk>',
            {
            'id': 1
            },
            format ='json'
        )
        self.assertEqual(response.status_code, status.HTTP_400_BAD_REQUEST)
        self.assertEqual(cohorte.id,'1')
        self.assertEqual(response.data['mesagge'], 'No se ha encontrado estudiante')
    def test_create_cohortet(self):
        cohorte = CohorteStructure().build cohorte JSON()
        response = self.client.post(
            self.url + '',
            cohorte,
            format='json')

        self.assertEqual(response.status_code, status.HTTP_201_CREATED)
```

Figura 5 Datos para Test Case Cohorte

4. Prueba Unitaria para registrar y buscar un estudiante

```python
class StudentStructure:

    def build_student_JSON(self):
        return{
            'name':"carla",
            'last_name':"troya",
            'identification':str(faker.random_number(digits=11)),
            'address': faker.email(),
            'cell_phone': faker.phone_number()
        }

    def create_student(self):
        return Student.objects.create(**self.build_student_JSON())
```

Figura 6 Datos para Test Case Student

```python
class StudentTestCase(TestSetUp):
    url= '/api/students/'
    def test_search_student(self):
        student = StudentStructure().create_student()
        response  = self.client.get(
            self.url + '<int:pk>',
        {
        }, format= 'json' )
        self.assertEqual(response.status_code, status.HTTP_200_OK)
        self.assertEqual(response.data['id'], student )
    def test search student error(self):
        student = StudentStructure().create_student()
        response = self.client.get(
            self.url + '<int:pk>',
            {
            'id': 1
            }, format ='json')
        self.assertEqual(response.status_code, status.HTTP_400_BAD_REQUEST)
        self.assertEqual(student.id,'1')
        self.assertEqual(response.data['mesagge'], 'No se ha encontrado estudiante')
    def test create student(self):
        student = StudentStructure().build_student_JSON()
        response = self.client.post(
            self.url + '',
            student,
            format='json'  )
        self.assertEqual(response.status code, status.HTTP 201 CREATED)
        self.assertDictEqual(Student.objects.all().count(), 1)
        self.assertEqual(response.data.student['id'], student.id)
```

Figura 7 Test Case de Student

5. Prueba Unitaria para buscar y crear matriculas

```
class EnrollmentStructure:

    def build_enrollment_JSON(self):
        return{
            'student': 1,
            'cohorte':1,
            'tipe_pay':1,
            'cuotas': 3,
            'day_limite': 2,
            'cash': 0,
            'discount': 12

        }

    def create_enrollment(self):
        return Enrollement.objects.create(**self.build_enrollment_JSON())
```

Figura 8 Test Case de Enrollment

```
class EnrollmentTestCase(TestSetUp):
    url= '/api/enrollment/'

    def test_search_enrollment(self):
        enrollment = EnrollmentStructure().create_enrollment()
        response  = self.client.get(
            self.url + '<int:pk>',
        {

        },
      zself.assertEqual(response.status_code, status.HTTP_200_OK)
        self.assertEqual(response.data['id'], enrollment )

    def test_search_enrollment_error(self):
        enrollment = EnrollmentStructure().create_enrollment()
        response = self.client.get(
            self.url + '<int:pk>',
            {
            'id': 1
            },
            format ='json'

        )
```

Figura 9 Test Case de Enrollment

6. Resultados de lo test realizados

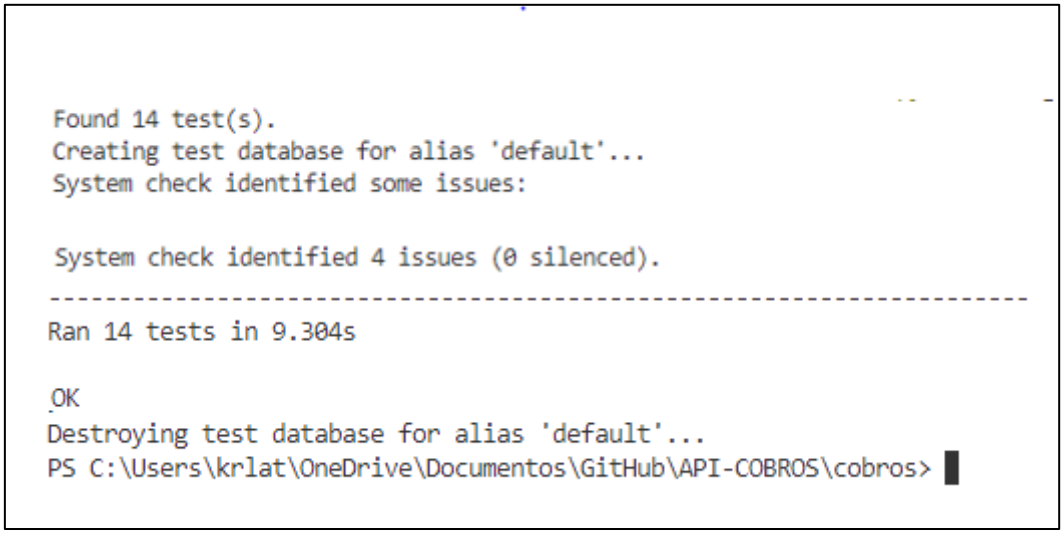


Figura 10 Resultado de los test en general

Pruebas de Funcionalidad

Pruebas de Funcionalidad

Proyecto: Desarrollo de un aplicativo web para la gestión de cobros del preuniversitario “CENES”

VALIDACIÓN DE LAS PRUEBAS DE CAJA NEGRA AL APLICATIVO WEB DE GESTIÓN DE COBROS DEL PREUNIVERSITARIO "CENES".

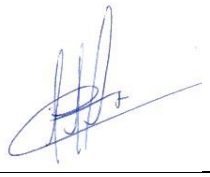
E = Encargado de la elaboración del documento.

R = Encargado de la revisión del documento.

A = Encargado de la aprobación del documento.

NOMBRE	CARGO	E	R	A	FIRMA
Carla Isabel Troya Capa	Tesista	X			Firmado electrónicamente por: CARLA ISABEL TROYA CAPA
Edwin René Guamán Quinche	Revisor Técnico		X	X	Firmado electrónicamente por: EDWIN RENE GUAMAN QUINCHE

<table>
<tr><td colspan="2">Requerimiento funcional</td><td colspan="5">RF 001</td></tr>
<tr><td colspan="2">Identificador: CP001</td><td colspan="5">Título: Iniciar sesión</td></tr>
<tr><td colspan="7">Detalle de la Prueba</td></tr>
<tr><td colspan="2">Fecha de realización:</td><td colspan="5">31-01-2023</td></tr>
<tr><td colspan="2">Requerimiento de la prueba:</td><td colspan="5">Iniciar Sesión</td></tr>
<tr><td colspan="2">Objetivo</td><td colspan="5">Evaluar y verificar que el módulo de inicio de sesión funcione de forma adecuada</td></tr>
<tr><td colspan="2">Descripción</td><td colspan="5">El propósito de esta prueba es verificar que el software cumple con el correcto inicio de sesión.</td></tr>
<tr><td colspan="2">Hardware requerido</td><td colspan="5">Computadora</td></tr>
<tr><td colspan="2">Software Requerido</td><td colspan="5">Navegador Web</td></tr>
<tr><td colspan="2">Criterios de éxito</td><td colspan="5">El usuario inicia sesión correctamente y es redirigido a la pantalla principal.</td></tr>
<tr><td colspan="2">Criterios de falla</td><td colspan="5">Capos vacíos o incorrectos, no permite el inicio de sesión y presenta un mensaje exponiendo el error.</td></tr>
<tr><td colspan="2">Precondiciones</td><td colspan="5">Ingresar a la dirección URL del sistema, donde se presenta la pantalla de inicio de sesión</td></tr>
<tr><td colspan="2">Procedimiento de prueba</td><td colspan="5">Llenar el formulario presentado.
Tener en cuenta que todos los datos son obligatorios</td></tr>
<tr><td colspan="2">Parámetros para el caso de prueba</td><td colspan="5">Nombre de usuario
Contraseña del usuario</td></tr>
<tr><td colspan="2">Resultado esperado</td><td colspan="5">Iniciar Sesión</td></tr>
<tr><td colspan="2" rowspan="6">Flujo del caso de prueba</td><td>N°</td><td>Usuario</td><td>N°</td><td>Sistema</td><td>Resultado</td></tr>
<tr><td>1</td><td>Ingresa en el formulario los datos</td><td></td><td></td><td></td></tr>
<tr><td>2</td><td>Clic en el botón Iniciar Sesión</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>3</td><td>Valida la información ingresada</td><td></td></tr>
<tr><td></td><td></td><td>4</td><td>Da acceso al usuario y muestra la página principal con los permisos de acuerdo al tipo usuario</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td>Sesión iniciada con éxito</td></tr>
<tr><td colspan="2">Resultado Obtenido</td><td colspan="5">Prueba Exitosa SI (X) NO ( )</td></tr>
<tr><td colspan="2" rowspan="2">Perfil de usuario</td><td colspan="2">Usuario</td><td colspan="2">Nombre</td><td>Firma</td></tr>
<tr><td colspan="2">Administrador</td><td colspan="2">Ing. Cristhian Salazar</td><td>  </td></tr>
</table>

	secretaria	Guadalupe Hurtado	
	estudiante	Anahí Montero	
Comentarios			

<table>
<tr><td colspan="2">Requerimiento funcional:</td><td colspan="5">RF 002</td></tr>
<tr><td colspan="2">Identificador: CP002</td><td colspan="5">Título: Crear Curso</td></tr>
<tr><td colspan="7">Detalle de la Prueba</td></tr>
<tr><td colspan="2">Fecha de realización:</td><td colspan="5">31-01-2023</td></tr>
<tr><td colspan="2">Requerimiento de la prueba:</td><td colspan="5">Crear un curso</td></tr>
<tr><td colspan="2">Objetivo</td><td colspan="5">Evaluar y verificar que el módulo de cursos funcione de forma adecuada</td></tr>
<tr><td colspan="2">Descripción</td><td colspan="5">El propósito de esta prueba es verificar que el software cumple con la correcta creación de un curso.</td></tr>
<tr><td colspan="2">Hardware requerido</td><td colspan="5">Computadora</td></tr>
<tr><td colspan="2">Software Requerido</td><td colspan="5">Navegador Web</td></tr>
<tr><td colspan="2">Criterios de éxito</td><td colspan="5">El curso se crea correctamente y se presenta un mensaje de éxito.</td></tr>
<tr><td colspan="2">Criterios de falla</td><td colspan="5">Capos vacíos, no permite crear el curso y presenta un mensaje exponiendo el error.</td></tr>
<tr><td colspan="2">Precondiciones</td><td colspan="5">Haber iniciado sesión previamente
En la lista desplegable, apartado Curso, seleccionar Crear</td></tr>
<tr><td colspan="2">Procedimiento de prueba</td><td colspan="5">Llenar el formulario presentado.
Tener en cuenta que todos los datos son obligatorios</td></tr>
<tr><td colspan="2">Parámetros para el caso de prueba</td><td colspan="5">Nombre del cuso
Descripción del curso</td></tr>
<tr><td colspan="2">Resultado esperado</td><td colspan="5">Crear un curso</td></tr>
<tr><td colspan="2" rowspan="7">Flujo del caso de prueba</td><td>N°</td><td>Usuario</td><td>N°</td><td>Sistema</td><td>Resultado</td></tr>
<tr><td>1</td><td>Ingresa en el formulario los datos</td><td></td><td></td><td></td></tr>
<tr><td>2</td><td>Clic en el botón Aceptar</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>3</td><td>Valida la información ingresada</td><td></td></tr>
<tr><td></td><td></td><td>4</td><td>Guarda los datos</td><td></td></tr>
<tr><td></td><td></td><td>5</td><td>Muestra un mensaje de confirmación</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td>Curso creado con éxito</td></tr>
<tr><td colspan="2">Resultado Obtenido</td><td colspan="5">Prueba Exitosa SI (X) NO ( )</td></tr>
<tr><td colspan="2" rowspan="2">Perfil de usuario</td><td colspan="2">Usuario</td><td colspan="2">Nombre</td><td>Firma</td></tr>
<tr><td colspan="2">Administrador</td><td colspan="2">Ing. Cristhian Salazar</td><td>  </td></tr>
<tr><td colspan="2">Comentarios</td><td colspan="5"></td></tr>
</table>

Requerimiento funcional	RF 002				
Identificador: CP003	**Título**: Gestionar cursos				
Detalle de la Prueba					
Fecha de realización	31-01-2023				
Requerimiento de la prueba	Editar un curso				
Objetivo	Evaluar y verificar que el módulo de cursos permita la correcta edición/actualización de un curso.				
Descripción	El propósito de esta prueba es comprobar que el software cumple con la correcta edición/actualización de un curso.				
Hardware requerido	Computadora				
Software Requerido	Navegador Web				
Criterios de éxito	El curso se edita correctamente y se presenta un mensaje de éxito.				
Criterios de falla	Capos vacíos o inválidos, no permite editar el curso y presenta un mensaje exponiendo el error.				
Precondiciones	Haber iniciado sesión previamente, tener creado al menos un curso En la lista de todos los cursos seleccionar “Editar” en el curso que desee hacerlo				
Procedimiento de prueba	Cambiar el valor de los datos que desee editar Tener en cuenta que todos los datos son obligatorios				
Parámetros para el caso de prueba	Nombre y Descripción del curso				
Resultado esperado	Editar un curso				
Flujo del caso de prueba	**N°**	**Usuario**	**N°**	**Sistema**	**Resultado**
	1	Clic en el botón “Editar” del curso que desee			
			2	Obtiene el id del curso de la tabla Curso	
			3	Buscar el curso a través del id	
			4	Carga los datos del curso en los campos de la interfaz editar curso	
	5	Cambia los valores en los campos que desea y da clic en el botón “Aceptar”			
			6	Valida la información ingresada	
			7	Actualiza los datos	
			8	Muestra un mensaje de confirmación	
					Curso editado con éxito
Resultado Obtenido	**Prueba Exitosa** SI (X) NO ( )				
Perfil de usuario	**Usuario**		**Nombre**		**Firma**

	Administrador	Ing. Cristhian Salazar	
Comentarios			

<table>
<tr><td colspan="2">Requerimiento funcional</td><td colspan="5">RF 002</td></tr>
<tr><td colspan="2">Identificador: CP004</td><td colspan="5">Título: Gestionar cursos</td></tr>
<tr><td colspan="7">Detalle de la Prueba</td></tr>
<tr><td colspan="2">Fecha de realización</td><td colspan="5">31-01-2023</td></tr>
<tr><td colspan="2">Requerimiento de la prueba</td><td colspan="5">Eliminar un curso</td></tr>
<tr><td colspan="2">Objetivo</td><td colspan="5">Evaluar y verificar que el módulo de cursos permita la correcta eliminación de un curso.</td></tr>
<tr><td colspan="2">Descripción</td><td colspan="5">El propósito de esta prueba es comprobar que el software cumple con la correcta eliminación de un curso.</td></tr>
<tr><td colspan="2">Hardware requerido</td><td colspan="5">Computadora</td></tr>
<tr><td colspan="2">Software Requerido</td><td colspan="5">Navegador Web</td></tr>
<tr><td colspan="2">Criterios de éxito</td><td colspan="5">El curso se elimina correctamente y se presenta un mensaje de éxito. Se podrá eliminar siempre y cuando no existan matrículas en este curso.</td></tr>
<tr><td colspan="2">Criterios de falla</td><td colspan="5">Existen matrículas registradas en el curso, no permite eliminar el curso y presenta un mensaje exponiendo el error.</td></tr>
<tr><td colspan="2">Precondiciones</td><td colspan="5">Haber iniciado sesión previamente, tener creado al menos un curso En la lista de todos los cursos seleccionar “Eliminar” en el curso que desee hacerlo</td></tr>
<tr><td colspan="2">Procedimiento de prueba</td><td colspan="5">Presionar el botón “Eliminar” del curso que desee.</td></tr>
<tr><td colspan="2">Parámetros para el caso de prueba</td><td colspan="5">Ninguno</td></tr>
<tr><td colspan="2">Resultado esperado</td><td colspan="5">Eliminar un curso</td></tr>
<tr><td colspan="2" rowspan="11">Flujo del caso de prueba</td><td>N°</td><td>Usuario</td><td>N°</td><td>Sistema</td><td>Resultado</td></tr>
<tr><td>1</td><td>Clic en el botón “eliminar” del curso que desee</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>Muestra un mensaje de confirmación</td><td></td></tr>
<tr><td>3</td><td>Da clic en el botón “Aceptar”</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>4</td><td>Buscar el curso y verifica que no tenga estudiantes matriculados</td><td></td></tr>
<tr><td></td><td></td><td>5</td><td>Elimina el curso</td><td></td></tr>
<tr><td></td><td></td><td>6</td><td>Recarga la tabla de cursos</td><td></td></tr>
<tr><td></td><td></td><td>7</td><td>Muestra un mensaje de confirmación</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td>Curso eliminado con éxito</td></tr>
<tr><td colspan="2">Resultado Obtenido</td><td colspan="5">Prueba Exitosa SI (X) NO ( )</td></tr>
</table>

Perfil de usuario	**Usuario**	**Nombre**	**Firma**
	Administrador	Ing. Cristhian Salazar	
Comentarios			

<table>
<tr><td colspan="2">Requerimiento funcional:</td><td colspan="5">RF 003</td></tr>
<tr><td colspan="2">Identificador: CP005</td><td colspan="5">Título: Gestionar Cohorte</td></tr>
<tr><td colspan="7">Detalle de la Prueba</td></tr>
<tr><td colspan="2">Requerimiento de la prueba:</td><td colspan="5">Crear una Cohorte</td></tr>
<tr><td colspan="2">Objetivo</td><td colspan="5">Evaluar y verificar que el módulo de cohorte funcione de forma adecuada</td></tr>
<tr><td colspan="2">Descripción</td><td colspan="5">El propósito de esta prueba es verificar que el software cumple con la correcta creación de una cohorte.</td></tr>
<tr><td colspan="2">Hardware requerido</td><td colspan="5">Computadora</td></tr>
<tr><td colspan="2">Software Requerido</td><td colspan="5">Navegador Web</td></tr>
<tr><td colspan="2">Criterios de éxito</td><td colspan="5">La cohorte se crea correctamente y se presenta un mensaje de éxito.</td></tr>
<tr><td colspan="2">Criterios de falla</td><td colspan="5">Capos vacíos o inválidos, no permite crear la cohorte y presenta un mensaje exponiendo el error.</td></tr>
<tr><td colspan="2">Precondiciones</td><td colspan="5">Haber iniciado sesión previamente
Crear al menos un curso
En la lista desplegable, apartado Cohorte, seleccionar Crear</td></tr>
<tr><td colspan="2">Procedimiento de prueba</td><td colspan="5">Llenar el formulario presentado.
Tener en cuenta que todos los datos son obligatorios</td></tr>
<tr><td colspan="2">Parámetros para el caso de prueba</td><td colspan="5">Seleccionar el curso
Nombre de la cohorte
Fecha de inicio
Fecha de fin
Costo a crédito
Costo en efectivo</td></tr>
<tr><td colspan="2">Resultado esperado</td><td colspan="5">Crear una cohorte</td></tr>
<tr><td colspan="2" rowspan="8">Flujo del caso de prueba</td><td>N°</td><td>Usuario</td><td>N°</td><td>Sistema</td><td>Resultado</td></tr>
<tr><td>1</td><td>Selecciona el curso en la lista desplegable</td><td></td><td></td><td></td></tr>
<tr><td>2</td><td>Ingresa los demás datos y da clic en el botón Aceptar</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>3</td><td>Valida la información ingresada</td><td></td></tr>
<tr><td></td><td></td><td>4</td><td>Guarda los datos</td><td></td></tr>
<tr><td></td><td></td><td>5</td><td>Muestra un mensaje de confirmación</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td>Curso creado con éxito</td></tr>
<tr><td colspan="2">Resultado Obtenido</td><td colspan="5">Prueba Exitosa SI (X) NO ( )</td></tr>
<tr><td colspan="2" rowspan="2">Perfil de usuario</td><td colspan="2">Usuario</td><td colspan="2">Nombre</td><td>Firma</td></tr>
<tr><td colspan="2">Administrador</td><td colspan="2">Ing. Cristhian Salazar</td><td>  </td></tr>
<tr><td colspan="2">Comentarios</td><td colspan="5"></td></tr>
</table>

Requerimiento funcional:	RF 003				
Identificador: CP006	**Título**: Gestionar Cohorte				
Detalle de la Prueba					
Fecha de realización	31-01-2023				
Requerimiento de la prueba	Editar una cohorte				
Objetivo	Evaluar y verificar que el módulo de cohorte permita la correcta edición/actualización de una cohorte.				
Descripción	El propósito de esta prueba es comprobar que el software cumple con la correcta edición/actualización de una cohorte.				
Hardware requerido	Computadora				
Software Requerido	Navegador Web				
Criterios de éxito	La cohorte se edita correctamente y se presenta un mensaje de éxito.				
Criterios de falla	Capos vacíos o inválidos, no permite editar la cohorte y presenta un mensaje exponiendo el error.				
Precondiciones	Haber iniciado sesión previamente, tener creado al menos una cohorte En la lista de todas las cohortes seleccionar “Editar” en la cohorte que desee				
Procedimiento de prueba	Cambiar el valor de los datos que desee editar Tener en cuenta que todos los datos son obligatorios				
Parámetros para el caso de prueba	Seleccionar el curso Nombre de la cohorte Fecha de inicio Fecha de fin Costo a crédito Costo en efectivo				
Resultado esperado	Editar una cohorte				
Flujo del caso de prueba	**N°**	**Usuario**	**N°**	**Sistema**	**Resultado**
	1	Clic en el botón “Editar” de la cohorte que desee			
			2	Obtiene el id de la cohorte de la tabla Cohorte	
			3	Buscar la cohorte a través del id	
			4	Carga los datos de la cohorte en los campos de la interfaz editar cohorte	
	5	Cambia los valores en los campos que desea y da clic en el botón “Aceptar”			
			6	Valida la información ingresada	
			7	Actualiza los datos	
			8	Muestra un mensaje de confirmación	

					Cohorte editada con éxito
Resultado Obtenido	**Prueba Exitosa** SI (X) NO ( )				
Perfil de usuario	**Usuario**		**Nombre**	**Firma**	
	Administrador		Ing. Cristhian Salazar		
Comentarios					

Requerimiento funcional	RF 003				
Identificador: CP007	**Título**: Gestionar Cohorte				
Detalle de la Prueba					
Fecha de realización	31-01-2023				
Requerimiento de la prueba	Eliminar una cohorte				
Objetivo	Evaluar y verificar que el módulo de cohorte permita la correcta eliminación de una cohorte.				
Descripción	El propósito de esta prueba es comprobar que el software cumple con la correcta eliminación de una cohorte.				
Hardware requerido	Computadora				
Software Requerido	Navegador Web				
Criterios de éxito	La cohorte se elimina correctamente y se presenta un mensaje de éxito. Se podrá eliminar siempre y cuando no existan matrículas en esta cohorte.				
Criterios de falla	Existen matrículas registradas en la cohorte, no permite eliminar y presenta un mensaje exponiendo el error.				
Precondiciones	Haber iniciado sesión previamente, tener creado al menos una cohorte En la lista de todas las cohortes seleccionar “Eliminar” en la cohorte que desee				
Procedimiento de prueba	Presionar el botón “Eliminar” de la cohorte que desee.				
Parámetros para el caso de prueba	Ninguno				
Resultado esperado	Eliminar una cohorte				
Flujo del caso de prueba	**N°**	**Usuario**	**N°**	**Sistema**	**Resultado**
	1	Clic en el botón “eliminar” de la cohorte que desee			
			2	Muestra un mensaje de confirmación	
	3	Da clic en el botón “Aceptar”			
			4	Buscar la cohorte y verifica que no tenga estudiantes matriculados	
			5	Elimina la cohorte	
			6	Recarga la tabla de cohortes	
			7	Muestra un mensaje de confirmación	
					Cohorte eliminada con éxito
Resultado Obtenido	**Prueba Exitosa** SI (X) NO ( )				
Perfil de usuario	**Usuario**	**Nombre**		**Firma**	

	Administrador	Ing. Cristhian Salazar	
Comentarios			

Requerimiento funcional:	RF 004				
Identificador: CP008	**Título**: Crear Estudiante				
Detalle de la Prueba					
Requerimiento de la prueba	Crear Estudiante				
Objetivo	Evaluar y verificar que el módulo de estudiantes permita la correcta creación de un estudiante.				
Descripción	El propósito de esta prueba es comprobar que el software cumple con la correcta creación de un estudiante.				
Hardware requerido	Computadora				
Software Requerido	Navegador Web				
Criterios de éxito	El estudiante se crea correctamente y se presenta un mensaje de éxito.				
Criterios de falla	Campos vacíos o datos incorrectos y presenta un mensaje exponiendo el error.				
Precondiciones	Haber iniciado sesión previamente En la lista desplegable, apartado Estudiante, seleccionar Crear				
Procedimiento de prueba	Llenar el formulario presentado. Tener en cuenta que todos los datos son obligatorios				
Parámetros para el caso de prueba	Nombre y Apellido Celular Dirección Cédula				
Resultado esperado	Crear un estudiante				
Flujo del caso de prueba	**N°**	**Usuario**	**N°**	**Sistema**	**Resultado**
	1	Ingresa los demás datos			
	2	Da clic en el botón Aceptar			
			3	Valida la información ingresada	
			4	Guarda los datos	
			5	Muestra un mensaje de confirmación	
					Estudiante creado con éxito
Resultado Obtenido	**Prueba Exitosa** SI (X) NO ( )				
Perfil de usuario	**Usuario**	**Nombre**	**Firma**		
	Administrador	Ing. Cristhian Salazar			
	secretaria	Guadalupe Hurtado			
Comentarios					

Requerimiento funcional:	RF 005				
Identificador: CP009	**Título**: Editar estudiante				
Detalle de la Prueba					
Fecha de realización	31-01-2023				
Requerimiento de la prueba	Editar un estudiante				
Objetivo	Evaluar y verificar que el módulo de estudiante permita la correcta edición/actualización de un estudiante.				
Descripción	El propósito de esta prueba es comprobar que el software cumple con la correcta edición/actualización de un estudiante.				
Hardware requerido	Computadora				
Software Requerido	Navegador Web				
Criterios de éxito	El estudiante se edita correctamente y se presenta un mensaje de éxito.				
Criterios de falla	Campos vacíos o inválidos, no permite editar el estudiante y presenta un mensaje exponiendo el error.				
Precondiciones	Haber iniciado sesión previamente, tener creado al menos un estudiante En la lista de todos los estudiantes seleccionar “Editar” en el estudiante que desee				
Procedimiento de prueba	Cambiar el valor de los datos que desee editar Tener en cuenta que todos los datos son obligatorios				
Parámetros para el caso de prueba	Nombre y Apellido Celular Dirección Cédula				
Resultado esperado	Editar un estudiante				
Flujo del caso de prueba	**N°**	**Usuario**	**N°**	**Sistema**	**Resultado**
	1	Clic en el botón “Editar” del estudiante que desee			
			2	Obtiene el id del estudiante de la tabla Estudiante	
			3	Buscar el estudiante a través del id	
			4	Carga los datos del estudiante en los campos de la interfaz editar estudiante	
	5	Cambia los valores en los campos que desea y da clic en el botón “Aceptar”			
			6	Valida la información ingresada	
			7	Actualiza los datos	
			8	Muestra un mensaje de confirmación	

					Estudiante editado con éxito
Resultado Obtenido	**Prueba Exitosa** SI (X) NO ( )				
Perfil de usuario	**Usuario**	**Nombre**	**Firma**		
	Administrador	Ing. Cristhian Salazar			
Comentarios					

Requerimiento funcional:	RF 006				
Identificador: CP010	**Título**: Matricular estudiantes				
Detalle de la Prueba					
Fecha de realización:	31-01-2023				
Requerimiento de la prueba:	Crear una matrícula				
Objetivo	Evaluar y verificar que el módulo de matrícula funcione de forma adecuada				
Descripción	El propósito de esta prueba es verificar que el software cumple con la correcta creación de una matrícula.				
Hardware requerido	Computadora				
Software Requerido	Navegador Web				
Criterios de éxito	La matrícula se crea correctamente y se presenta un mensaje de éxito.				
Criterios de falla	Capos vacíos o inválidos, no permite crear la matrícula y presenta un mensaje exponiendo el error.				
Precondiciones	Haber iniciado sesión previamente Crear al menos un curso, una cohorte y un estudiante En la lista desplegable, apartado Matrícula, seleccionar Crear				
Procedimiento de prueba	Llenar el formulario presentado. Tener en cuenta que todos los datos son obligatorios				
Parámetros para el caso de prueba	Seleccionar la cohorte Seleccionar el estudiante Seleccionar tipo de pago Costo en efectivo/costo en cuotas (depende del tipo de pago) Día de pago descuento				
Resultado esperado	Crear una matrícula				
Flujo del caso de prueba	**N°**	**Usuario**	**N°**	**Sistema**	**Resultado**
	1	Selecciona el botón “Estudiante”			
	2	Seleccionar/Buscar el estudiante			
			3	Muestra datos del estudiante seleccionado	
	4	Selecciona el botón “Cohorte”			
	5	Seleccionar/Buscar la Cohorte a la que se va a matricular.			
			6	Muestra los datos de la Cohorte seleccionada	
	7	Seleccionar el tipo de pago			
			8	Activa el campo según el tipo de	

				pago seleccionado	
	9	Ingresa el valor de los campos restantes			
	10	Clic en Aceptar			
			11	Valida la información ingresada	
			12	Guarda los datos	
			13	Muestra un mensaje de confirmación	
					Matrícula creada con éxito
Resultado Obtenido	**Prueba Exitosa** SI ( ) NO (X)				
Perfil de usuario	**Usuario**		**Nombre**		**Firma**
	Administrador		Ing. Cristhian Salazar		
	secretaria		Guadalupe Hurtado		
Comentarios	En el caso de seleccionar el pago en efectivo, el usuario solicito que por defecto se ponga el valor del curso. Para pasar la prueba, se implementó en el método enrollment, el campo de valor del curso en efectivo para que se muestre el valor del curso en el campo efectivo.				

Requerimiento funcional:	RF 006				
Identificador: CP011	**Título**: Matricular Estudiantes				
Detalle de la Prueba					
Fecha de realización	31-01-2023				
Requerimiento de la prueba	Eliminar una matrícula				
Objetivo	Evaluar y verificar que el módulo de matrícula permita la correcta eliminación de una matrícula.				
Descripción	El propósito de esta prueba es comprobar que el software cumple con la correcta eliminación de una matrícula.				
Hardware requerido	Computadora				
Software Requerido	Navegador Web				
Criterios de éxito	La matrícula se elimina correctamente y se presenta un mensaje de éxito. Al eliminar la matrícula se eliminarán los pagos en el caso de existir.				
Criterios de falla	No permite eliminar y presenta un mensaje exponiendo el error.				
Precondiciones	Haber iniciado sesión previamente, tener creado al menos una matrícula En la lista de todas las matrículas seleccionar “Eliminar” en la matrícula que desee				
Procedimiento de prueba	Presionar el botón “Eliminar” de la matrícula que desee.				
Parámetros para el caso de prueba	Ninguno				
Resultado esperado	Eliminar una matrícula				
Flujo del caso de prueba	**N°**	**Usuario**	**N°**	**Sistema**	**Resultado**
	1	Clic en el botón “eliminar” de la matrícula que desee			
			2	Muestra un mensaje de confirmación	
	3	Da clic en el botón “Aceptar”			
			4	Buscar la matrícula	
			5	Elimina la matrícula	
			6	Recarga la tabla de matrículas	
			7	Muestra un mensaje de confirmación	
					Matrícula eliminada con éxito
Resultado Obtenido	**Prueba Exitosa** SI (X) NO ( )				
Perfil de usuario	**Usuario**	**Nombre**		**Firma**	

	Administrador	Ing. Cristhian Salazar	
Comentarios			

Requerimiento funcional:	RF 007				
Identificador: CP012	**Título**: Registrar un pago				
Detalle de la Prueba					
Fecha de realización	31-01-2023				
Requerimiento de la prueba	Registrar un pago				
Objetivo	Evaluar y verificar que el módulo de pagos permita la correcta creación de un pago.				
Descripción	El propósito de esta prueba es comprobar que el software cumple con la correcta creación de un pago.				
Hardware requerido	Computadora				
Software Requerido	Navegador Web				
Criterios de éxito	El pago se registra correctamente y se presenta un mensaje de éxito.				
Criterios de falla	Campos vacíos o datos incorrectos y presenta un mensaje exponiendo el error.				
Precondiciones	Haber iniciado sesión previamente En la lista desplegable, apartado Matrículas, buscar el estudiante al que se le registrara el pago y seleccionar la opción “Registrar Pago”				
Procedimiento de prueba	Llenar el formulario presentado. Tener en cuenta que todos los datos son obligatorios				
Parámetros para el caso de prueba	Fecha de pago Estado de pago				
Resultado esperado	Pago Registrado				
Flujo del caso de prueba	**N°**	**Usuario**	**N°**	**Sistema**	**Resultado**
			1	Busca todos los estudiantes matriculados	
			2	Carga la lista de estudiantes en la tabla “Pagos”	
	3	Selecciona un estudiante y da clic en el botón “Pagar”.			
			4	Muestra la interfaz registrar Pago	
	5	Ingresar la fecha del pago			
	6	Cambiar el estado de pago a “PAGADO”			
	7	Clic en Guardar			

			3	Valida la información ingresada	
			4	Guarda los datos	
			5	Muestra un mensaje de confirmación	
					Pago registrado con éxito

Resultado Obtenido	**Prueba Exitosa** SI ( ) NO (X)		
Perfil de usuario	**Usuario**	**Nombre**	**Firma**
	Administrador	Ing. Cristhian Salazar	
	secretaria	Guadalupe Hurtado	
Comentarios	El usuario estuvo de acuerdo con la interfaz brindada, pero sugirió que en la tabla de pagos por cuotas se le agregue el saldo resultante después de pagar cada cuota. Se solucionó colocando en la tabla de amortización un campo llamado saldo, donde se llama al valor del curso y se le va restando la cantidad mensual de cuotas hasta llegar a un saldo de 0.		

Requerimiento funcional:	RF 008				
Identificador: CP013	**Título**: Editar un Pago				
Detalle de la Prueba					
Fecha de realización	31-01-2023				
Requerimiento de la prueba	Editar un pago				
Objetivo	Evaluar y verificar que el módulo de pago permita la correcta edición/actualización de un pago.				
Descripción	El propósito de esta prueba es comprobar que el software cumple con la correcta edición/actualización de un pago.				
Hardware requerido	Computadora				
Software Requerido	Navegador Web				
Criterios de éxito	El pago se edita correctamente y se presenta un mensaje de éxito.				
Criterios de falla	Campos vacíos o inválidos, no permite editar el pago y presenta un mensaje exponiendo el error.				
Precondiciones	Haber iniciado sesión previamente, tener registrado al menos un pago En la lista de todos los pagos seleccionar “Editar” en el pago que desee realizar la acción				
Procedimiento de prueba	Cambiar el valor de los datos que desee editar Tener en cuenta que todos los datos son obligatorios				
Parámetros para el caso de prueba	Fecha de pago Estado de pago				
Resultado esperado	Editar un pago				
Flujo del caso de prueba	**N°**	**Usuario**	**N°**	**Sistema**	**Resultado**
	1	Clic en el botón “Editar” del pago que desee			
			2	Activa las casillas del pago que se pueden editar	
	3	Editar el valor del campo			
	4	Da clic en el botón “Aceptar			
			5	Valida la información ingresada	
			6	Actualiza los datos	
			7	Muestra un mensaje de confirmación	
					Pago editado con éxito
Resultado Obtenido	**Prueba Exitosa** SI (X) NO ( )				
Perfil de usuario	**Usuario**		**Nombre**		**Firma**

	Administrador	Ing. Cristhian Salazar	
Comentarios			

Requerimiento funcional:	RF 009				
Identificador: CP014	**Título**: Ver Reporte				
Detalle de la Prueba					
Fecha de realización	31-01-2023				
Requerimiento de la prueba	Ver Reporte				
Objetivo	Evaluar y verificar que el módulo de reportes permita la correcta visualización de un reporte.				
Descripción	El propósito de esta prueba es comprobar que el software cumple con la correcta visualización de un reporte.				
Hardware requerido	Computadora				
Software Requerido	Navegador Web				
Criterios de éxito	El reporte se puede visualizar sin errores.				
Criterios de falla	No se puede visualizar el reporte y presenta un mensaje exponiendo el error.				
Precondiciones	Haber iniciado sesión previamente, tener registrado al menos un curso/cohorte, matricula, estudiante y pago				
Procedimiento de prueba	En la lista desplegable seleccionar la opción ver reporte Se presentará una tabla con los datos del reporte, la cual tendrá opción de búsqueda				
Parámetros para el caso de prueba	Se puede buscar el: Estado de pago, Estudiante, Curso/Cohorte				
Resultado esperado	Visualizar reporte				
Flujo del caso de prueba	**N°**	**Usuario**	**N°**	**Sistema**	**Resultado**
			1	Busca todos los estudiantes matriculados	
			2	Carga la lista de estudiantes en la tabla “Reporte”	
	3	Ingresa en la barra de búsqueda el dato que desee buscar			
	4	Da clic en el botón “Buscar”			
			5	Busca el dato ingresado	
			6	En la interfaz el sistema mostrará una tabla con todos los estudiantes que contengan el dato que fue ingresado en la búsqueda	
	7	Visualiza los estudiantes que tiene relación con el dato ingresado			
					Se

					visualiza el reporte con éxito
Resultado Obtenido	**Prueba Exitosa** SI (X) NO ( )				
Perfil de usuario	**Usuario**	**Nombre**	**Firma**		
	Administrador	Ing. Cristhian Salazar			
Comentarios					

Requerimiento funcional:	RF 010				
Identificador: CP014	**Título**: Visualizar Matrículas				
Detalle de la Prueba					
Fecha de realización	31-01-2023				
Requerimiento de la prueba	Visualizar matrículas por parte del estudiante				
Objetivo	Evaluar y verificar que el módulo de matrículas permita la correcta visualización de la matrícula de cada estudiante.				
Descripción	El propósito de esta prueba es comprobar que el software cumple con la correcta visualización de las matrículas por parte del estudiante.				
Hardware requerido	Dispositivo móvil				
Software Requerido	Aplicación móvil				
Criterios de éxito	Las matrículas conjuntamente con los pagos de cada estudiante se pueden visualizar sin errores.				
Criterios de falla	No se puede visualizar las matrículas y presenta un mensaje exponiendo el error.				
Precondiciones	Haber iniciado sesión previamente, estar matriculado al menos en un curso/cohorte.				
Procedimiento de prueba	En la barra inferior seleccionar la opción Mis Cursos				
Parámetros para el caso de prueba	Ninguno				
Resultado esperado	Visualizar reporte				
Flujo del caso de prueba	**N°**	**Usuario**	**N°**	**Sistema**	**Resultado**
			1	Busca todos los cursos donde se encuentre matriculado el estudiante que inicio sesión	
			2	Carga la lista de cursos en la tabla “Cursos”, de la interfaz Mis Cursos	
	3	Visualiza los cursos donde se encuentra matriculado			
	4	Selecciona un curso			
			5	Obtiene el id del curso	
			6	Buscar el curso y sus pagos a través del id	
			7	Carga los datos de los pagos en los campos de la interfaz Pagos	
	8	Visualiza las			

		matrículas con sus pagos			
					Se visualiza las matrículas con éxito
Resultado Obtenido	**Prueba Exitosa** SI (X) NO ( )				
Perfil de usuario	**Usuario**	**Nombre**	**Firma**		
	Administrador	Ing. Cristhian Salazar			
	Estudiante 1	Anahí Montero			
	Estudiante 2	Juan Ochoa			
Comentarios					

Pruebas de Carga y Estrés

Pruebas de Carga y Estrés

Proyecto: Desarrollo de un aplicativo web para la gestión de cobros del preuniversitario “CENES”

VALIDACIÓN DE LAS PRUEBAS DE CARGA Y ESTRÉS PARA EL APLICATIVO WEB DE GESTIÓN DE COBROS DEL PREUNIVERSITARIO "CENES".

E = Encargado de la elaboración del documento.

R = Encargado de la revisión del documento.

A = Encargado de la aprobación del documento de pruebas de carga y estrés.

NOMBRE	CARGO	E	R	A	FIRMA
Carla Isabel Troya Capa	Tesista	X			Firmado electrónicamente por: CARLA ISABEL TROYA CAPA
Edwin René Guamán Quinche	Revisor Técnico		X	X	Firmado electrónicamente por: EDWIN RENE GUAMAN QUINCHE

Pruebas de Carga y Estrés

La aplicación “CENES” se encuentra alojada en servidor Centos 7 de 30 GB de espacio y 3072 MB de RAM con 200% de CPU (2V Core).

Para la ejecución de las pruebas de carga y estrés se utilizó la herramienta JMeter que permitió diseñar, ejecutar y ver en distintas formas el resultado de la ejecución del plan de pruebas. Estas pruebas fueron realizadas por la tesista a la aplicación que se encuentra ya alojada en un servidor con la siguiente dirección:

https://proeditsclub.com/cobros/frontend/index.html#/auth/login

A continuación, se describen los pasos para realizar las pruebas de carga y estrés en JMeter.

1. Se define el nombre el proyecto para la prueba de la aplicación “CENES”, como se muestra en la Figura 1.

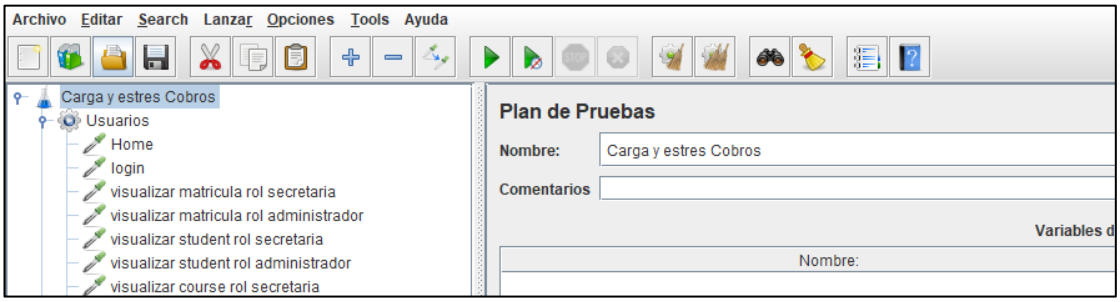


Figura 1 Creación de la prueba de Carga y Estrés

2. Crear el Grupo de Hilos donde se designó un nombre, numero de hilos (número de peticiones), el periodo de subida en segundos (tiempo de ejecución) y el contador del bucle (puede ser de 1 o sin fin), como se muestra en la Figura 2.

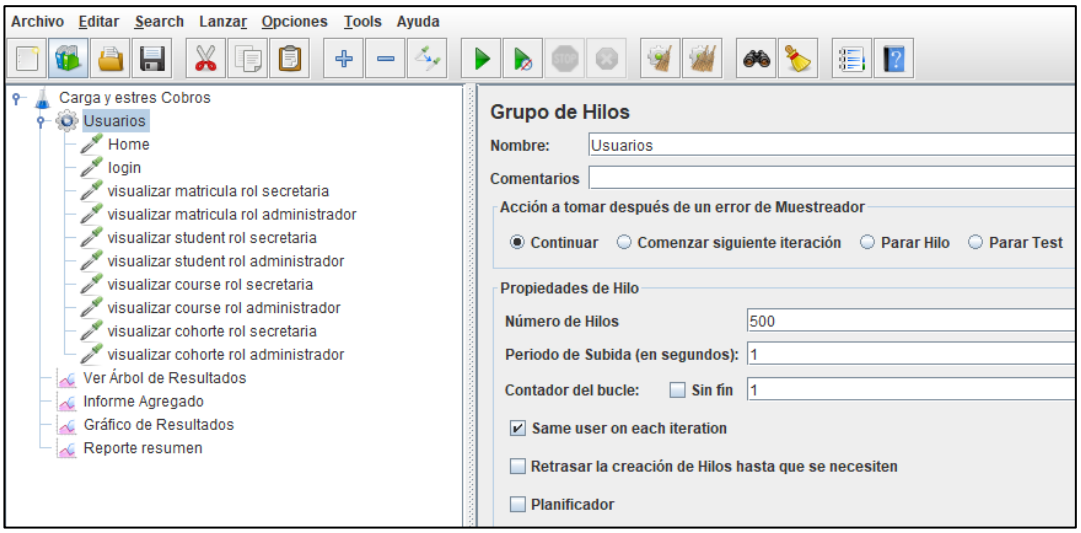


Figura 2 Creación de Grupo de Hilos para las pruebas

3. Crear la Petición HTTP, se tuvo a consideración realizar pruebas para las peticiones Get y Post que se estima que podrían producir cuellos de botella, para la configuración de las peticiones se pone un nombre a la petición, definimos el protocolo (Http, Https), nombre del servidor o dirección IP, tipo de petición (POST, GET), y la ruta de la petición a realizar, como se muestra en la figura 3.

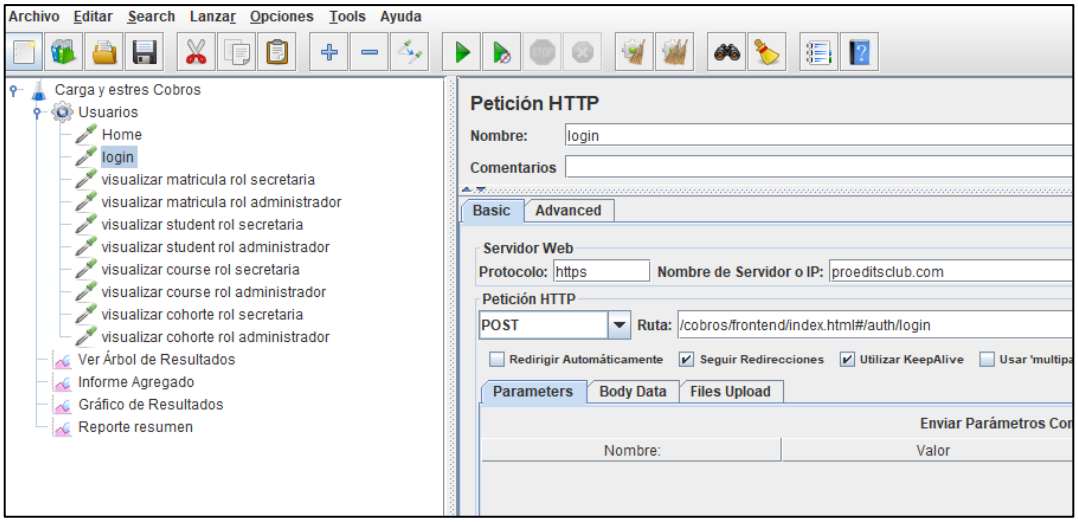


Figura 3 Creación de peticiones

4. Añadimos la vista para mostrar los resultados en modo árbol, informe agregado, grafico de resultados y resumen reporte. En la figura 4 se muestran los resultados en modo árbol donde se puede observar que todas peticiones realizadas pasan el test de prueba.

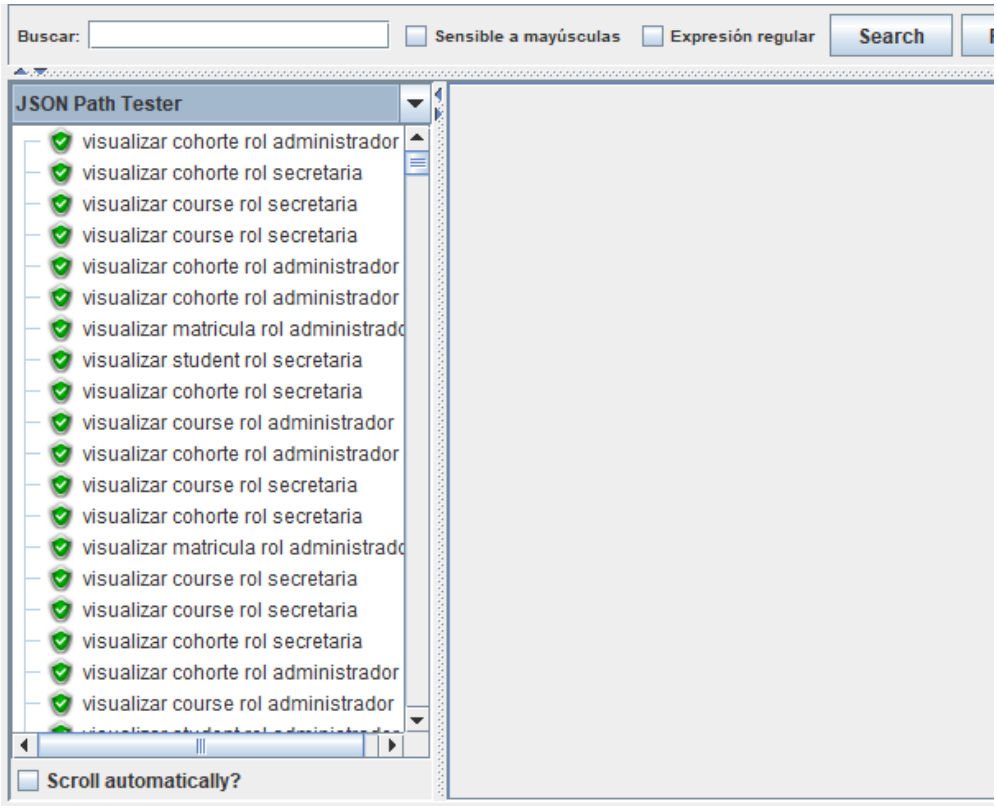


Figura 4 Peticiones realizada en la prueba de carga y estrés

5. En la figura 5 se muestran el informe agregado donde se puede observar las peticiones realizadas, el porcentaje de erro el rendimiento y otras demás características.

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec
Home	500	4415	3139	11112	11492	14335	747	26070	6,00%	18,3/sec	96,53	2,46
login	500	1755	940	3918	5440	12077	2	23988	0,60%	18,5/sec	101,27	4,15
visualizar ...	500	1908	1138	3946	5308	7706	1	23293	1,40%	18,7/sec	102,01	2,65
visualizar ...	500	1115	818	2198	3059	5112	3	9362	0,20%	19,3/sec	105,59	2,76
visualizar s...	500	1140	802	1839	3532	4843	181	7333	0,00%	20,4/sec	112,18	2,93
visualizar s...	500	947	711	2037	3140	3600	181	10832	0,00%	20,7/sec	113,54	2,97
visualizar c...	500	785	566	1681	1978	3731	186	5083	0,00%	21,2/sec	116,63	3,05
visualizar c...	500	691	532	1345	1716	3386	179	4255	0,00%	21,5/sec	118,27	3,09
visualizar c...	500	606	469	818	1462	3473	180	6035	0,00%	23,0/sec	126,08	3,30
visualizar c...	500	489	430	691	869	2962	180	4373	0,00%	24,2/sec	133,17	3,48
Total	5000	1385	726	3249	4891	11124	1	26070	0,82%	168,9/sec	922,64	25,42

Figura 5 Informe de la prueba de carga y estrés

6. En la figura 6 se muestran el grafico de resultados donde se observa como la aplicación "CENES" al procesar las 500 peticiones por segundo logra mantener un rendimiento estable a lo largo de la ejecución de la prueba.

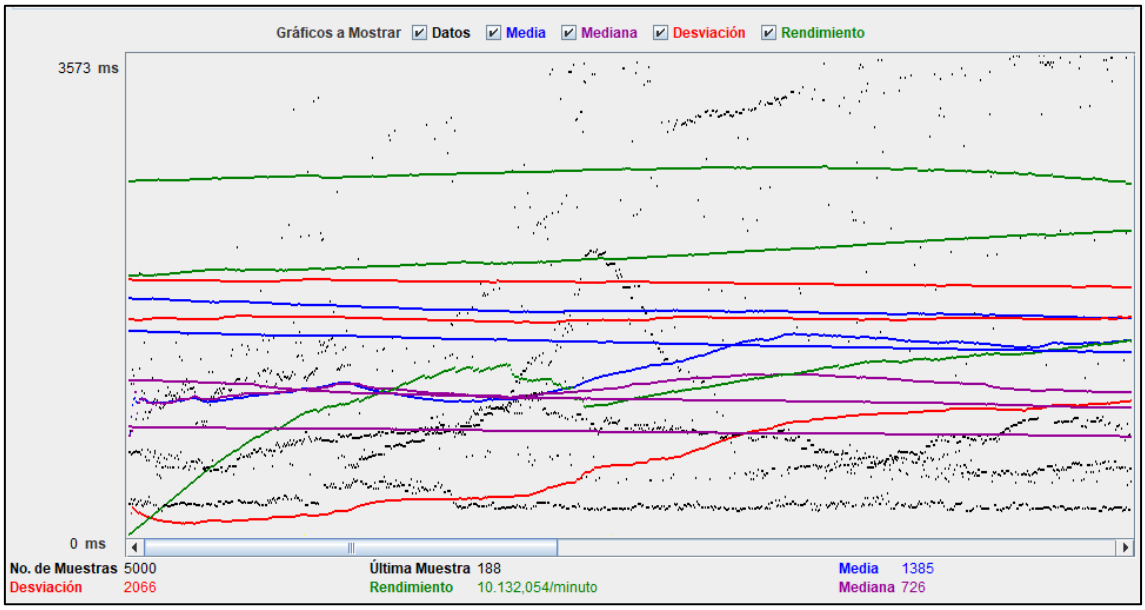


Figura 6 Resultado de la prueba de carga y estrés

7. En la figura 7 se puede observar el resumen de la prueba realizada obteniendo un 0,08% de error, es decir la aplicación tiende a fallar en lo minino al momento de visualizar las matrículas y los estudiantes, por otro lado, se obtuve un rendimiento del 210%, evidenciando que la aplicación puede soportar 500 peticiones por segundo, por lo tanto, se concluye que la aplicación funciona de manera correcta en un minino de respuesta de 1milisengundo y en un máximo de respuestas en 17466 milisegundos.

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Bytes
Home	500	4415	747	26070	4090,16	6,00%	18,3/sec	96,53	2,46	5411,7
login	500	1755	2	23988	2361,59	0,60%	18,5/sec	101,27	4,15	5602,8
visualizar mat...	500	1908	1	23293	1937,64	1,40%	18,7/sec	102,01	2,65	5574,5
visualizar mat...	500	1115	3	9362	1017,40	0,20%	19,3/sec	105,59	2,76	5616,9
visualizar stu...	500	1140	181	7333	995,74	0,00%	20,4/sec	112,18	2,93	5624,0
visualizar stu...	500	947	181	10832	963,25	0,00%	20,7/sec	113,54	2,97	5624,0
visualizar cou...	500	785	186	5083	722,97	0,00%	21,2/sec	116,63	3,05	5624,0
visualizar cou...	500	691	179	4255	598,19	0,00%	21,5/sec	118,27	3,09	5624,0
visualizar coh...	500	606	180	6035	656,85	0,00%	23,0/sec	126,08	3,30	5623,3
visualizar coh...	500	489	180	4373	458,02	0,00%	24,2/sec	133,17	3,48	5623,3
Total	5000	1385	1	26070	2066,49	0,82%	168,9/sec	922,64	25,42	5594,8

Figura 7 Informe final de la prueba de carga y estrés

Anexo 10. Manual de Instalación del Aplicativo web COBROS

1. Creación del Api en el servidor

Paso 1: Entrar a la cuenta de LucusHost para acceder a la cuenta de cPanel, tal y como se observa en la figura 1.

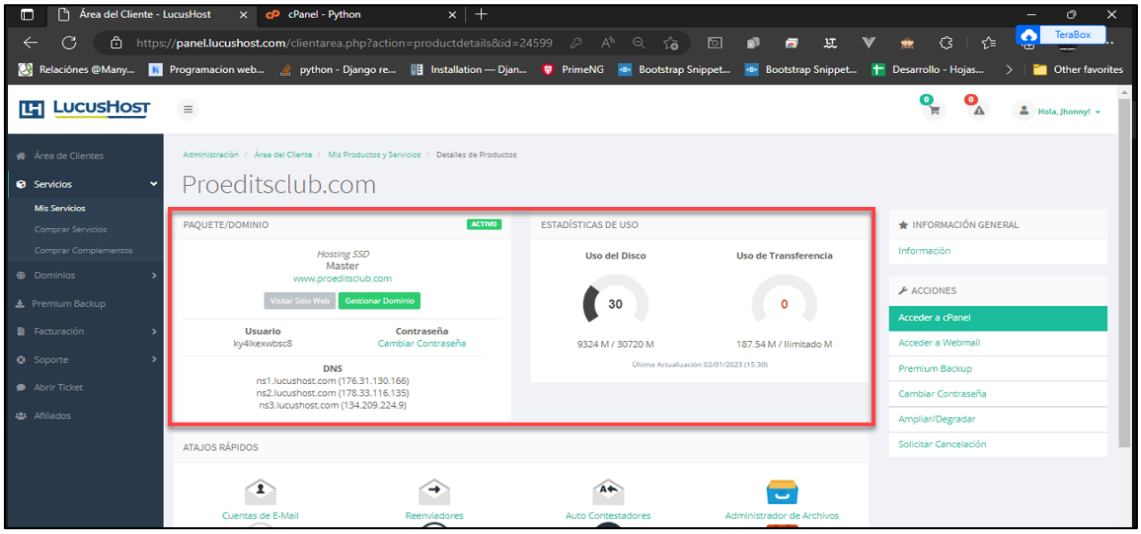


Figura 1 Acceso a LucusHost

Paso 2: Hacer clic en Acceder a cPanel como se muestra en la figura 2.

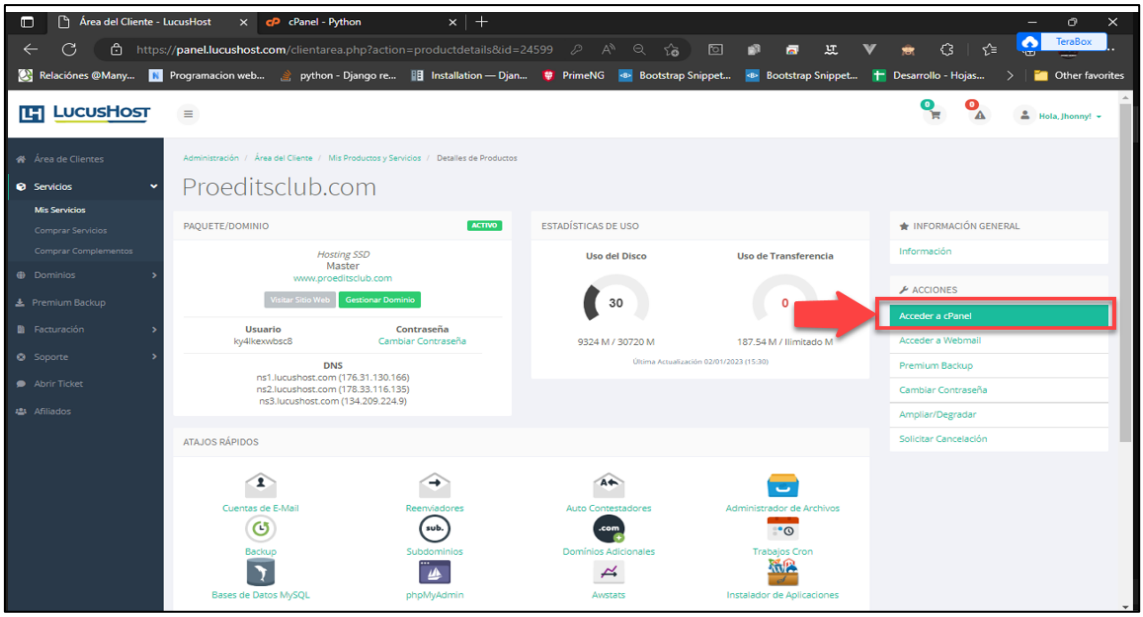


Figura 2 Acceder a CPanel

Paso 3: En sistema muestra la página principal de cPanel y acceder a la sección de Software como se muestra en la Figura 3.

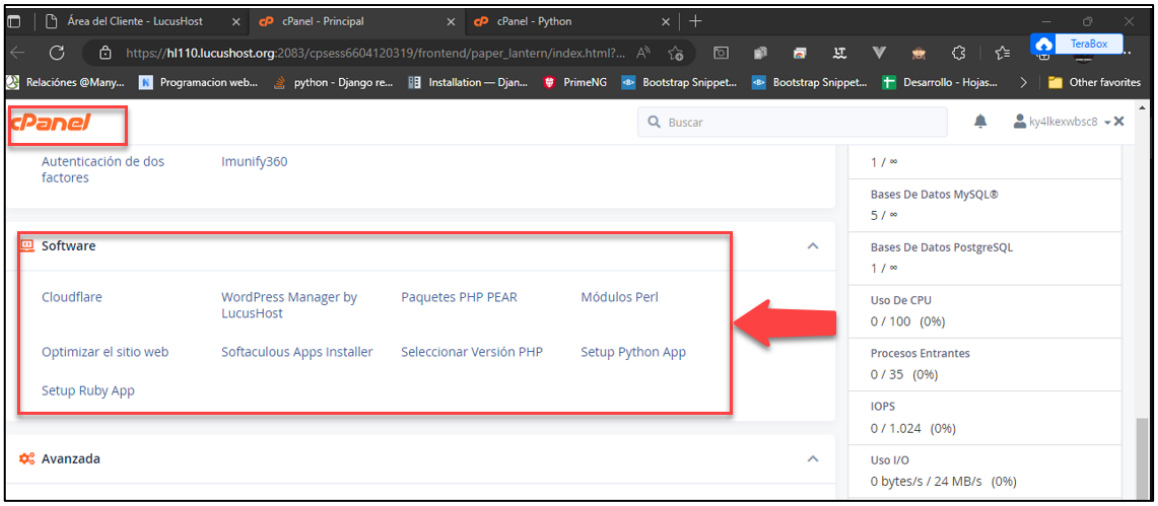


Figura 3 Página cPanel

Paso 4: Hacer clic en **"stup Python"** como se muestra en la Figura 4.

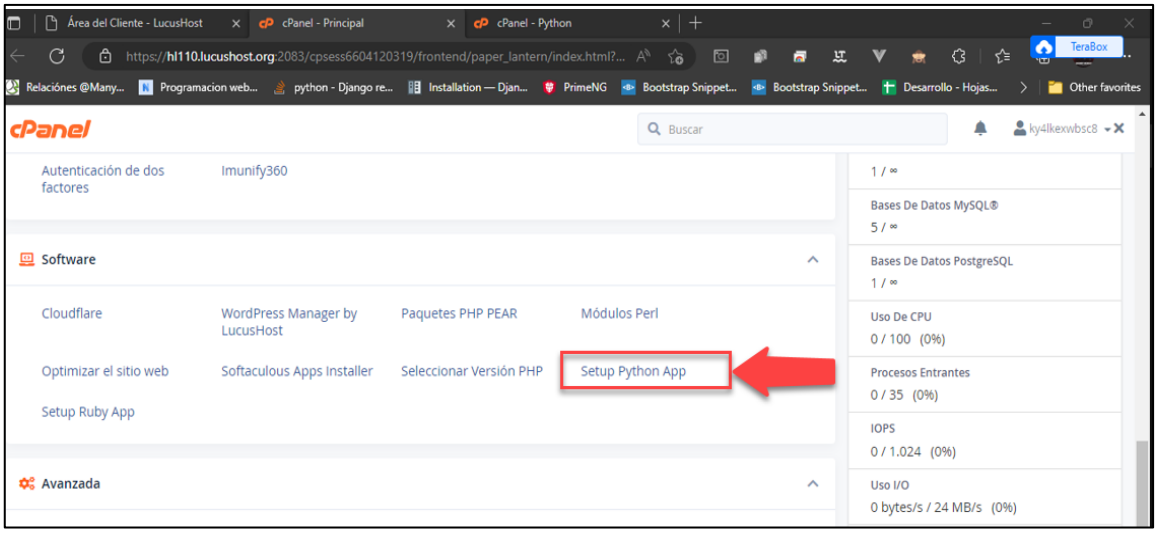


Figura 4 Página cPanel

Paso 5: hacer clic en "Create Aplication" como se muestra en la Figura 5

Figura 5 Crear aplicación en Python

Paso 6: Llenar los campos correctamente para crear la aplicación para el Api y hacer clic en **Create** como se muestra en la Figura 6.

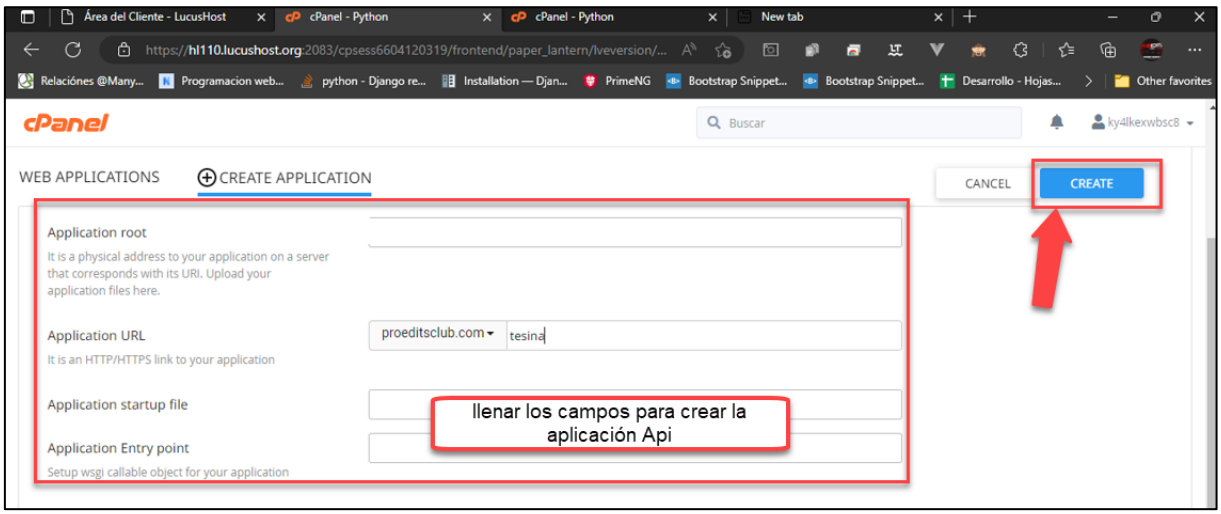


Figura 6 Crear aplicación para el Api

Paso 7: Finalmente tenemos la aplicación como se muestra en la Figura 7.

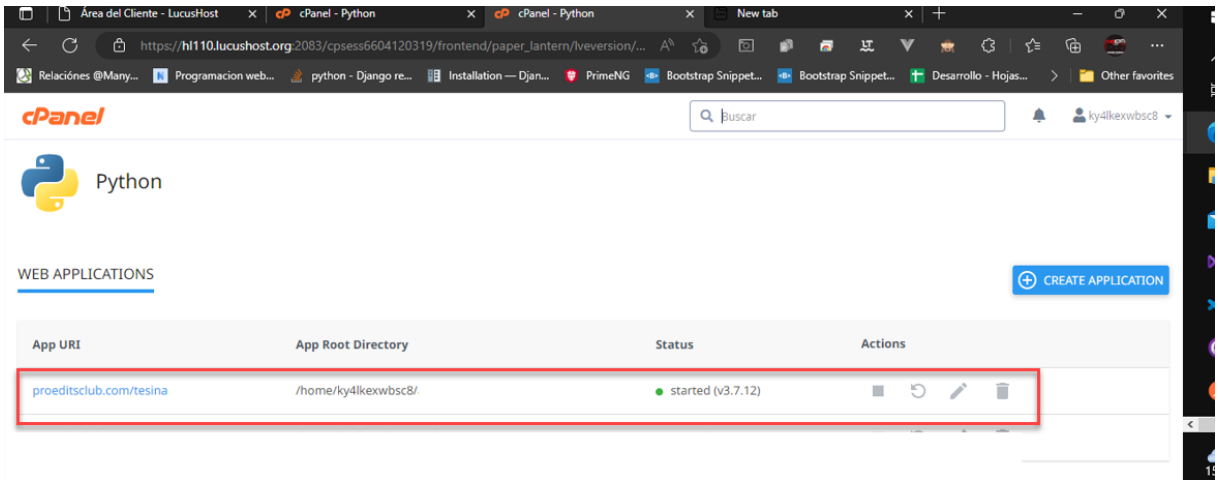


Figura 7 Aplicación creada

Paso 8: Validar la creación exitosa ingresando al dominio de la aplicación https://proeditsclub.com/cobros/ como se muestra en la Figura 8.

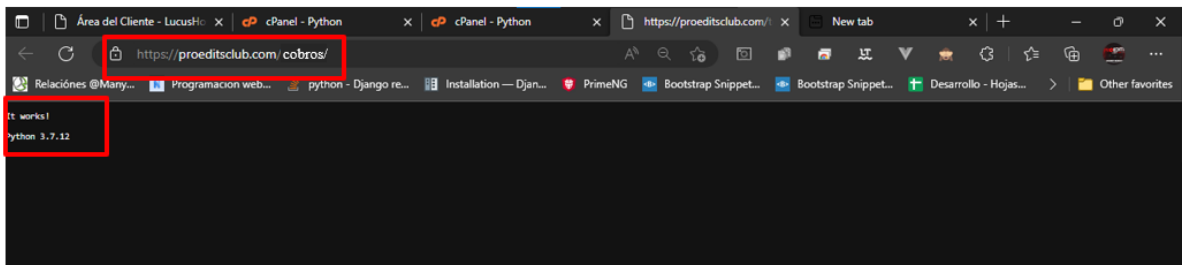


Figura 8 Validar la aplicación creada

2. Configuración e Instalación de Django y sus Dependencias

Paso 1: Acceder a la aplicación creada y copiar la ruta que se genera, como se muestra en la Figura 9.

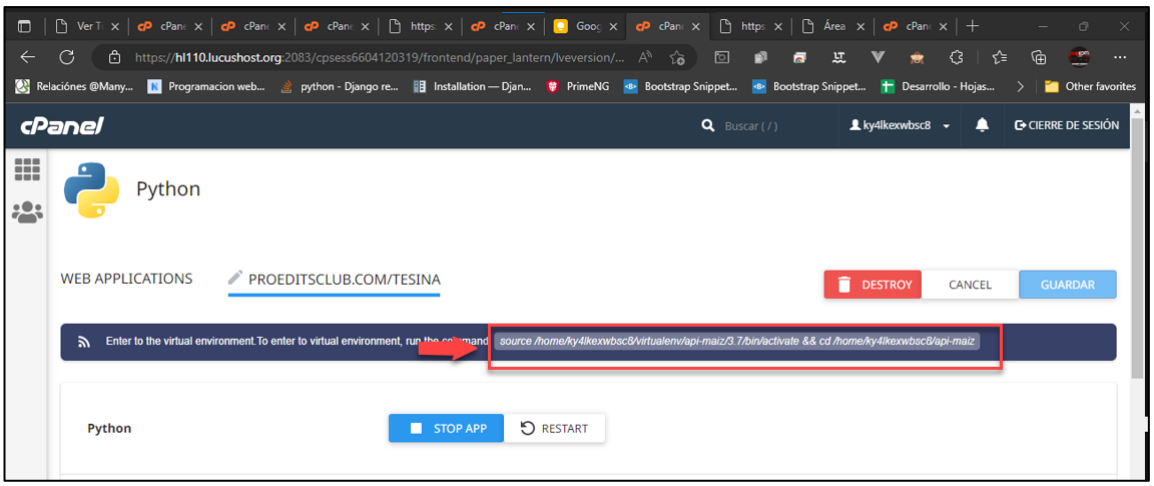


Figura 9 Ruta generada en la aplicación creada

Paso 2: Acceder a la terminal y pegar la ruta, como se muestra en la Figura 10.

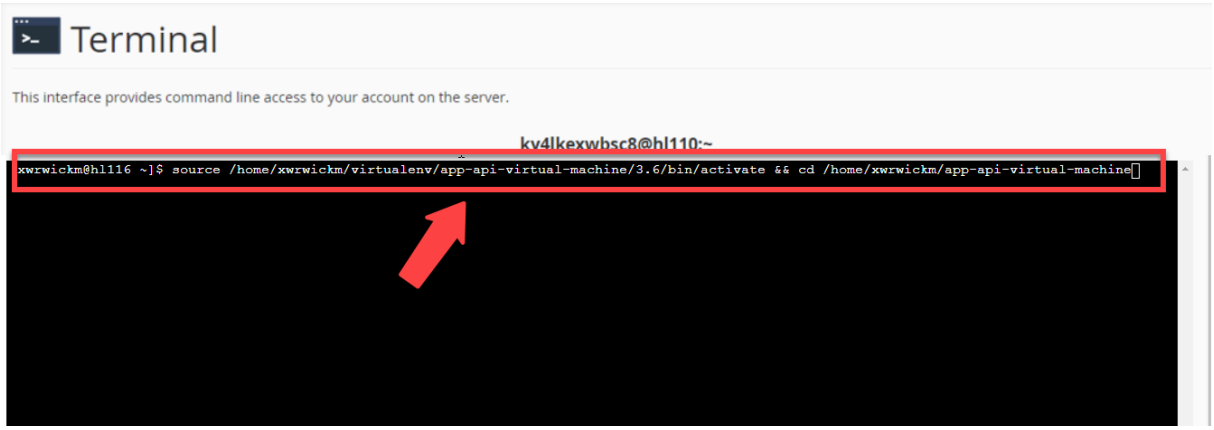


Figura 10 Pegar la ruta de la aplicación en el terminal

Paso 3: Instalar Django con el siguiente comando: **pip install django**

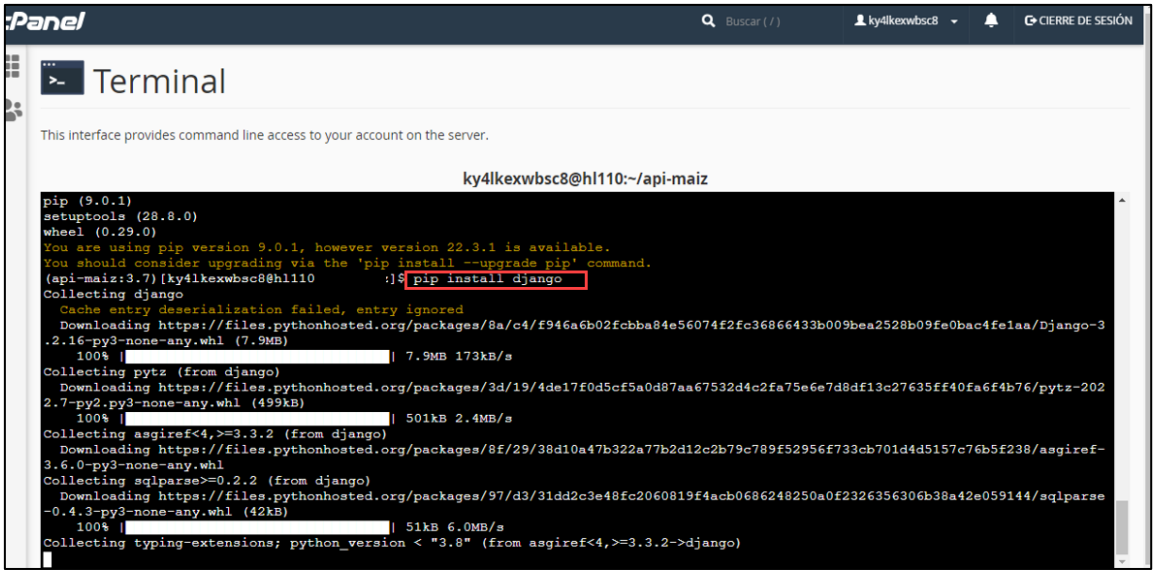


Figura 11 Instalación de django

Paso 4: Instalar asgiref con el siguiente comando: "**pip install asgiref"**

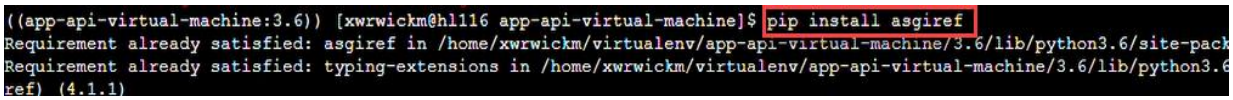


Figura 12 Instalación de asgiref

Paso 5: Instalar awsebcli con el siguiente comando: pip install awsebcli

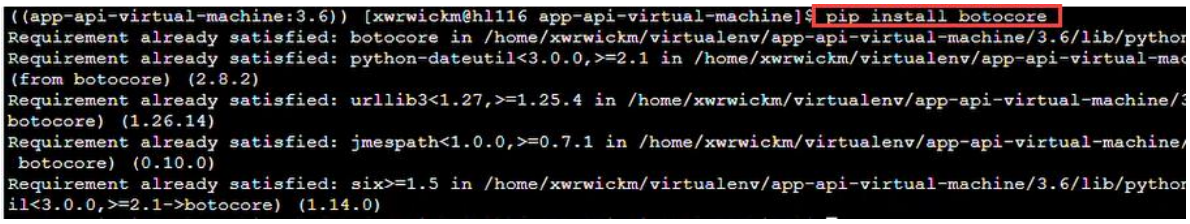


Figura 13 Instalación de awsebcli

Paso 6: Instalar certifi con el siguiente comando: pip install certifi

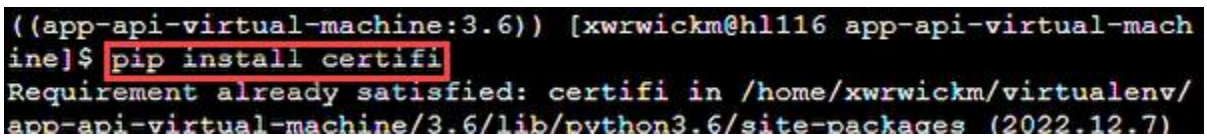


Figura 14 Instalación de certifi

Paso 7: Instalar charset-normalizer con el siguiente comando: pip install charset-normalizer

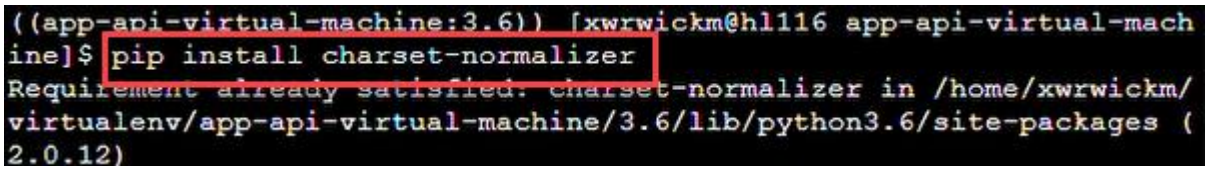


Figura 15 Instalación de charset-normalizer

Paso 8: Instalar djangorestframework con el siguiente comando: pip install djangorestframework.

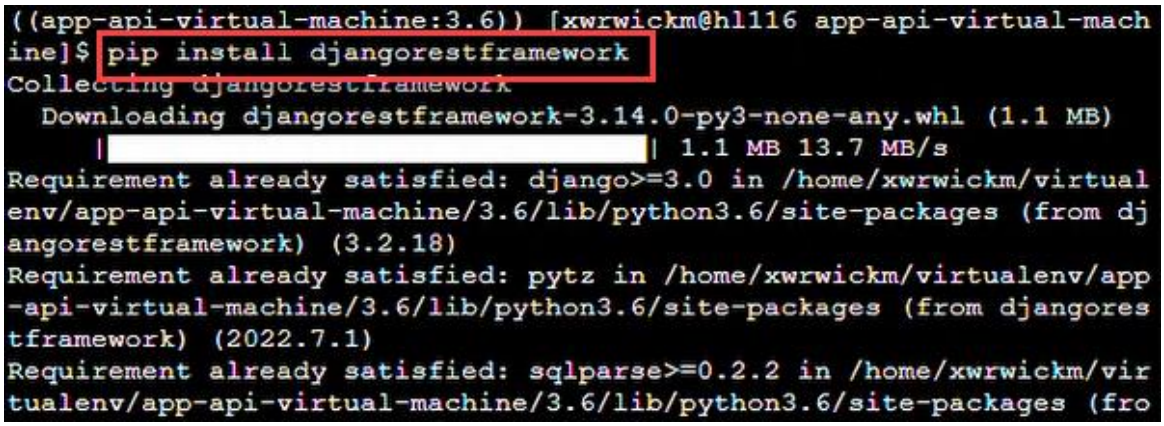


Figura 16 Instalación de djangorestframework

Paso 9: Instalar filelock con el siguiente comando: pip install filelock

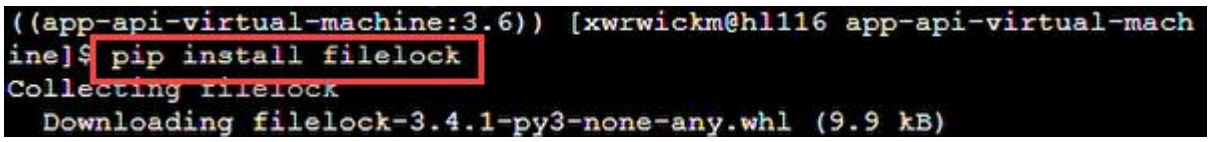


Figura 17 Instalación de filelock

Paso 10: Instalar future con el siguiente comando: pip install future

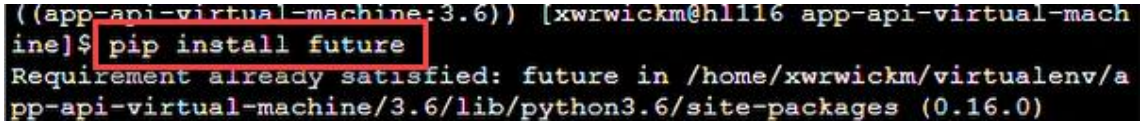


Figura 18 Instalación de future

Paso 11: Instalar idna con el siguiente comando: pip install idna

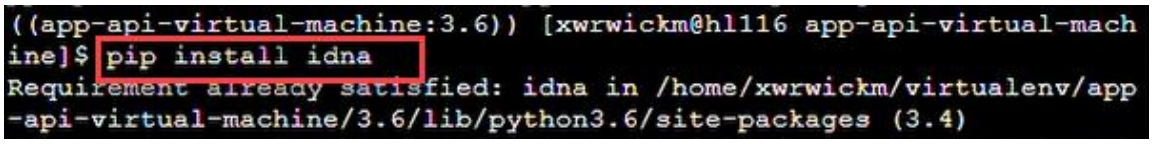


Figura 19 Instalación de idna

Paso 12: Instalar jmespath con el siguiente comando: pip install jmespath

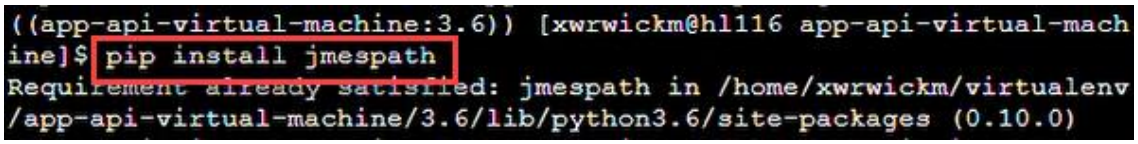


Figura 20 Instalación de jmespath

Paso 13: Instalar python-dateutil con el siguiente comando: pip install python-dateutil

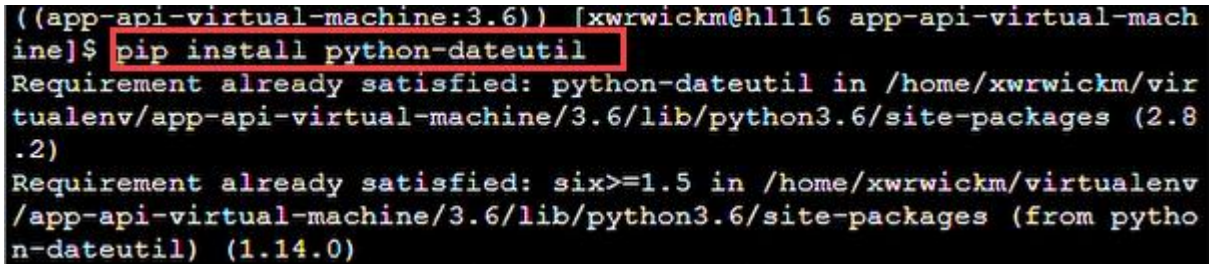


Figura 21 Instalación de python-dateutil

Paso 14: Instalar pytz con el siguiente comando: pip install pytz

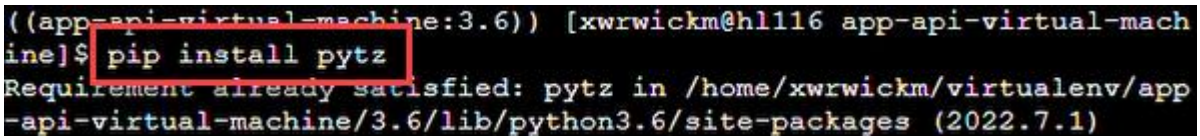


Figura 22 Instalación de pytz

Paso 15: Instalar PyYAML con el siguiente comando: pip install PyYAML

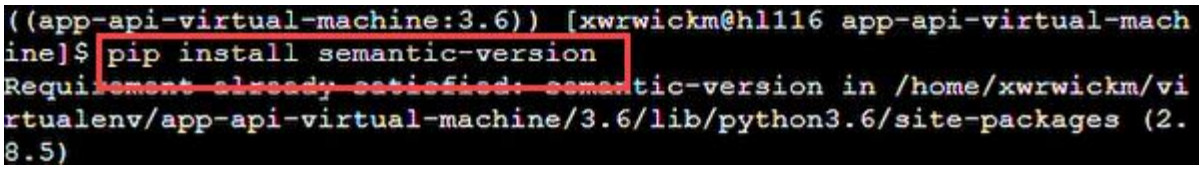


Figura 23 Instalación de PyYAML

Paso 16: Instalar termcolor con el siguiente comando: pip install termcolor

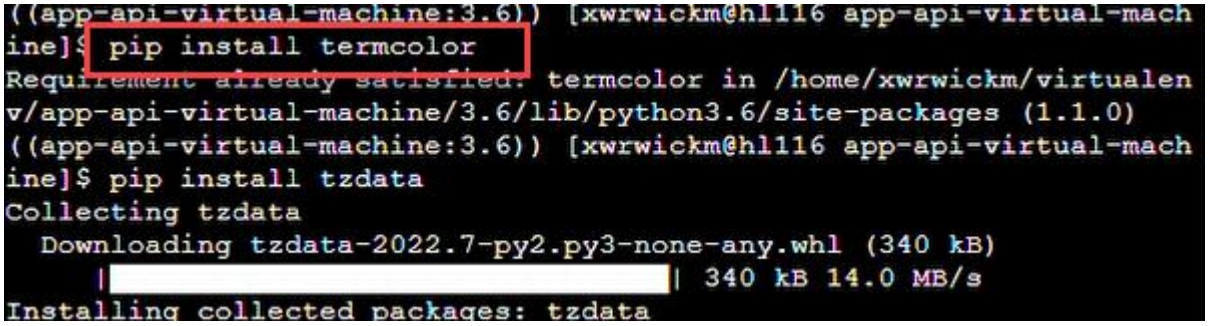


Figura 24 Instalación de termcolor

Paso 17: Instalar tzdata con el siguiente comando: pip install tzdata

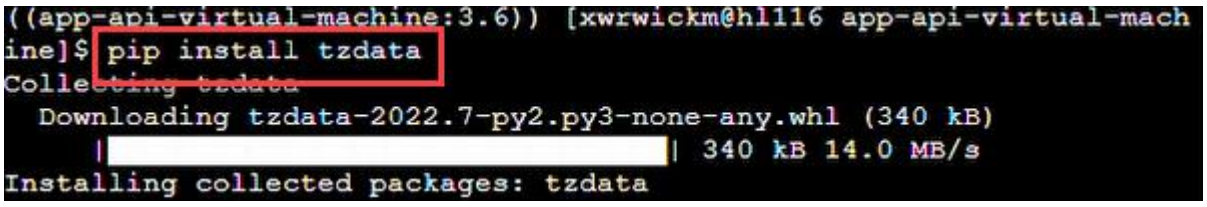


Figura 25 Instalación de tzdata

Paso 18: Instalar urllib3con el siguiente comando: pip install urllib3

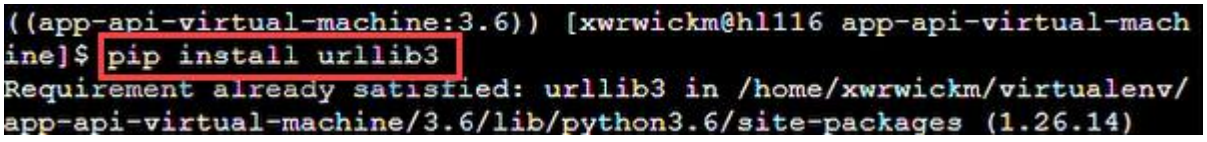


Figura 26 Instalación de urllib3

Paso 19: Instalar virtualenv con el siguiente comando: pip install virtualenv

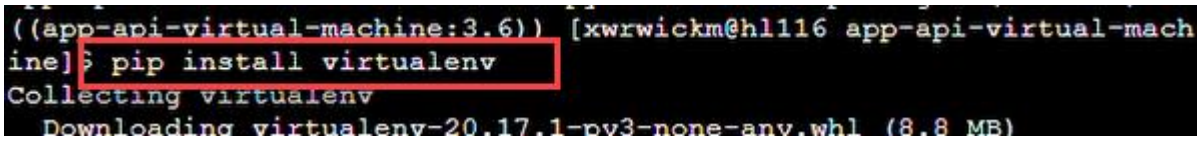


Figura 27 Instalación de virtualenv

Paso 20: Instalar wcwidth con el siguiente comando: pip install wcwidth

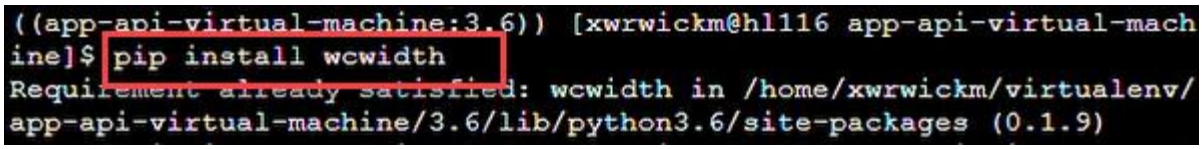


Figura 28 Instalación de wcwidth

Paso 21: Instalar whitenoise con el siguiente comando: pip install whitenoise

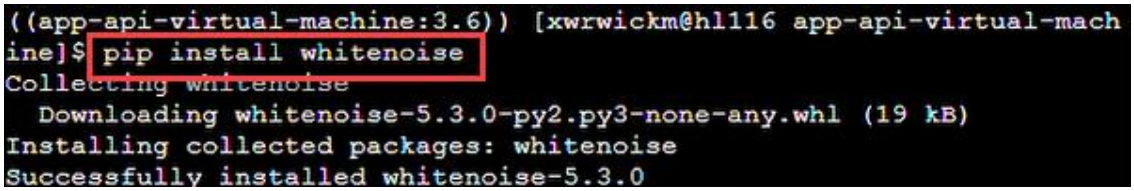


Figura 29 Instalación de whitenoise

3. Subir el API al servidor.

Paso 1: Abrir el programa WinSCP realizar la conexión con el servidor para transferir los archivos hacia el servidor como se muestra en la Figura 30.

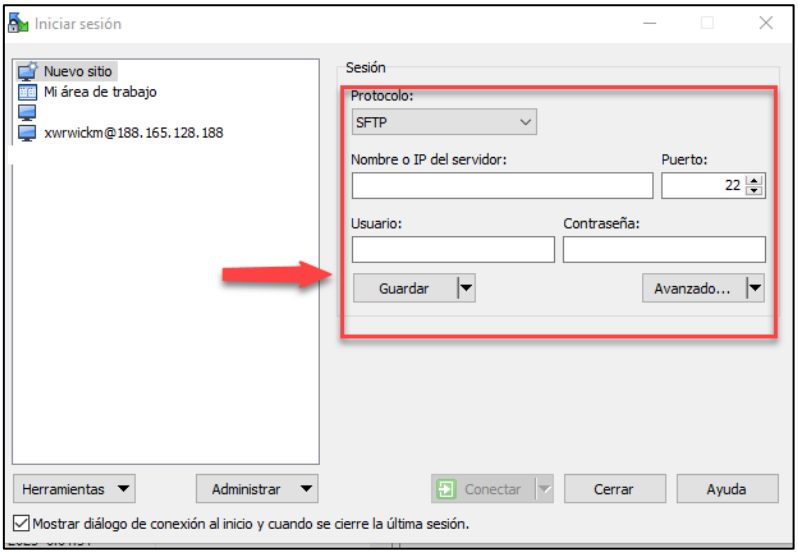


Figura 30 WinSCP para la conexión con el servidor

Paso 2: Acceder a la aplicación creada en el servidor para la transferencia de archivos como se muestra en la Figura 31.

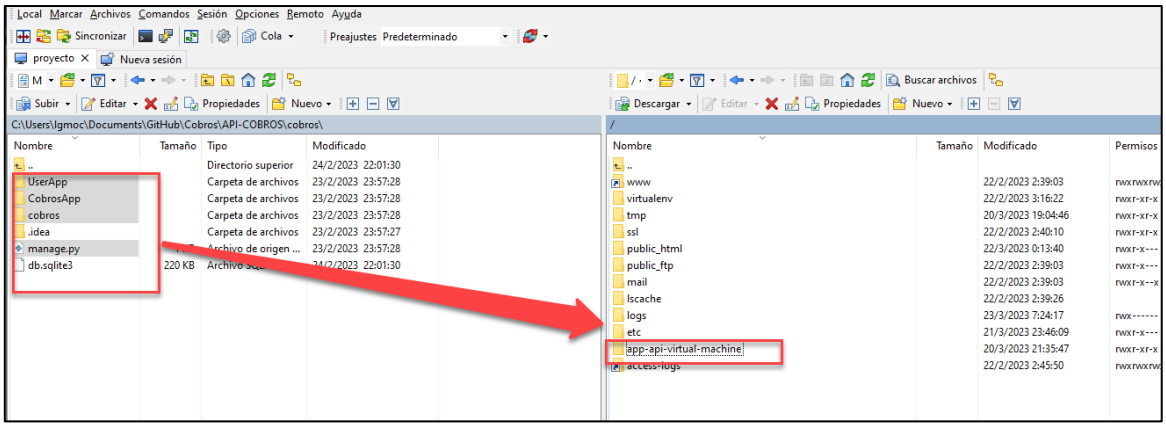


Figura 31 Transferir archivos api al servidor

Paso 3: Acceder al archivo **"passenger_wsgi.py"** como se muestra en la Figura 32.

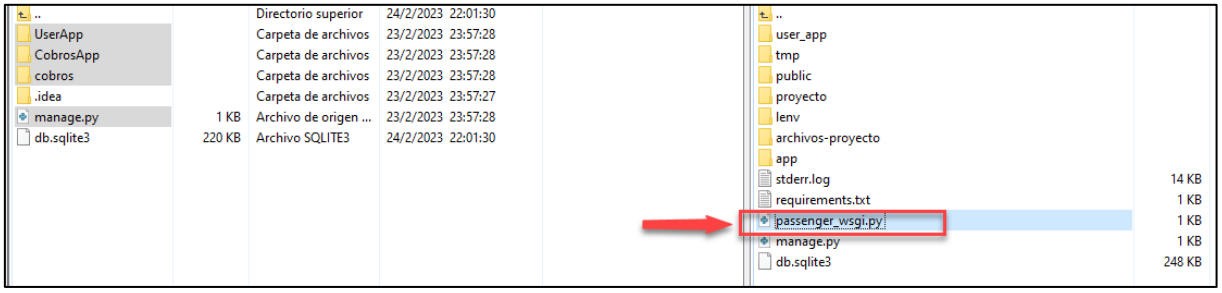


Figura 32 Acceder al archivo "passenger_wsgi.py"

Paso 4: Pegar la siguiente línea de comando *from proyecto.wsgi import application* en el archivo **"passenger_wsgi.py",** como se muestra en Figura 33.

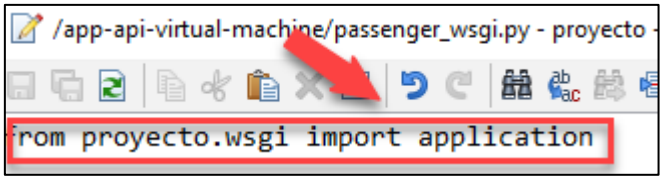


Figura 33 Modificando el archivo "passenger_wsgi.py"

Paso 5: Comprobar el funcionamiento del Api en el servidor mediante la Terminal con el siguiente comando: **"python manage.py run server".**

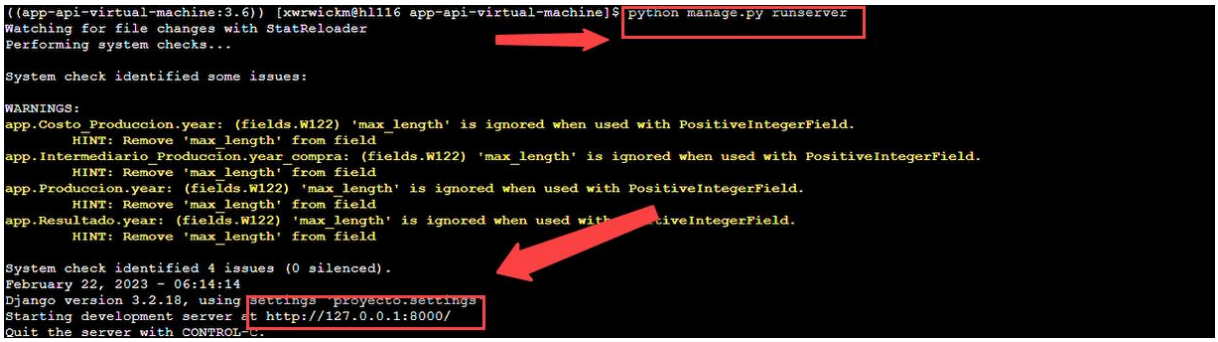


Figura 34 Funcionamiento del api mediante la Terminal

Paso 6: Comprobar el funcionamiento del api en la ruta del servidor: https://proeditsclub.com/cobros/api/

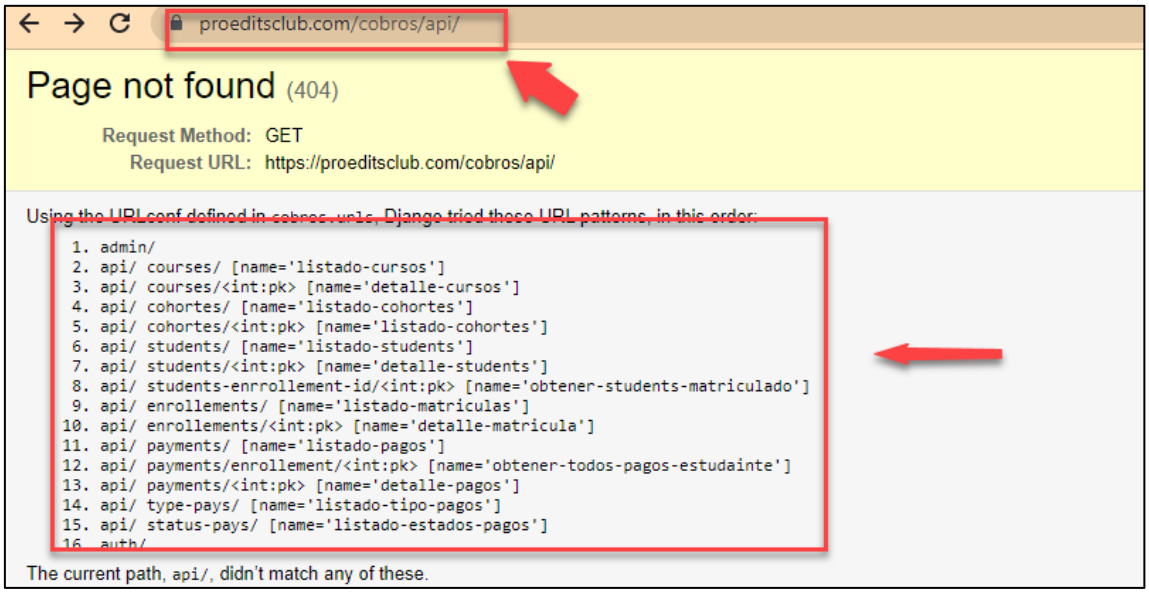


Figura 35 Comprobación del funcionamiento del api en el servidor

4. Consumir el Api desde el servidor Web.

Paso 1: Acceder a la Carpeta environments del Frontend, como se muestra en la Figura 36.

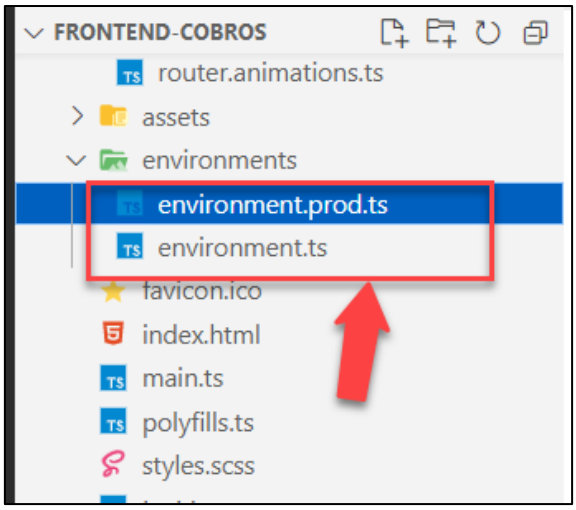


Figura 36 Carpeta Environments.

Paso 2: Acceder al archivo **environment.prod.ts** y realizar la conexión con el backend como se muestra en la Figura 37.

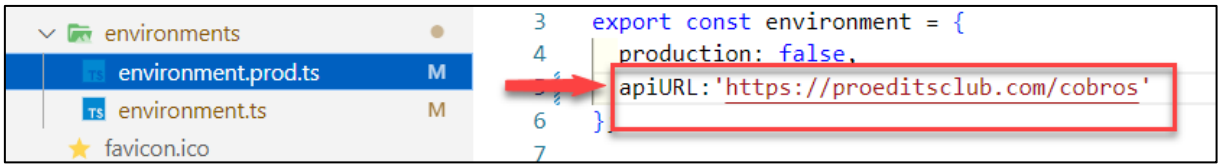


Figura 37 Archivo environment.prod.ts

Paso 3: Acceder al archivo **environment.ts** y realizar la conexión con el backend como se muestra en la Figura 38.

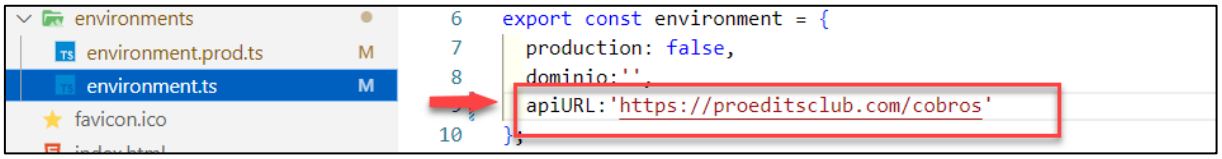


Figura 38 Archivo environment.ts

Paso 4: Generar los archivos del frontend mediante el comando correspondiente como se muestra en la Figura 39.

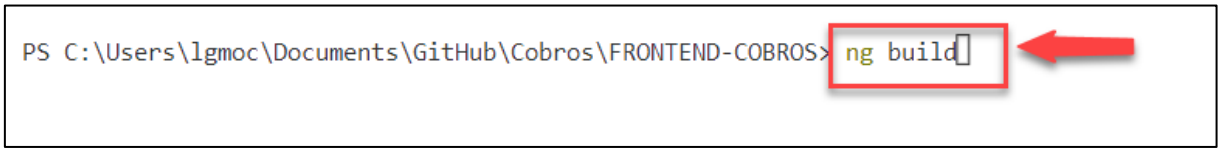


Figura 39 comando ng build

Paso 5: Acceder a la aplicación creada en el servidor mediante el programa WinSCP para la transferencia de archivos como se muestra en la Figura 40.

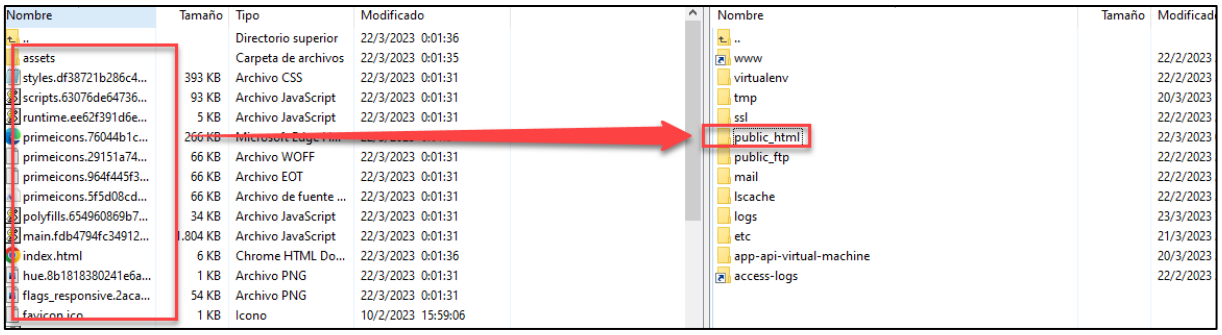


Figura 40 Acceder al archivo public_html para transferir los archivos del frontend

Paso 6: Comprobar el funcionamiento del api en la ruta del servidor: https://proeditsclub.com/cobros/frontend/index.html#/auth/login, como se muestra en la Figura 41.

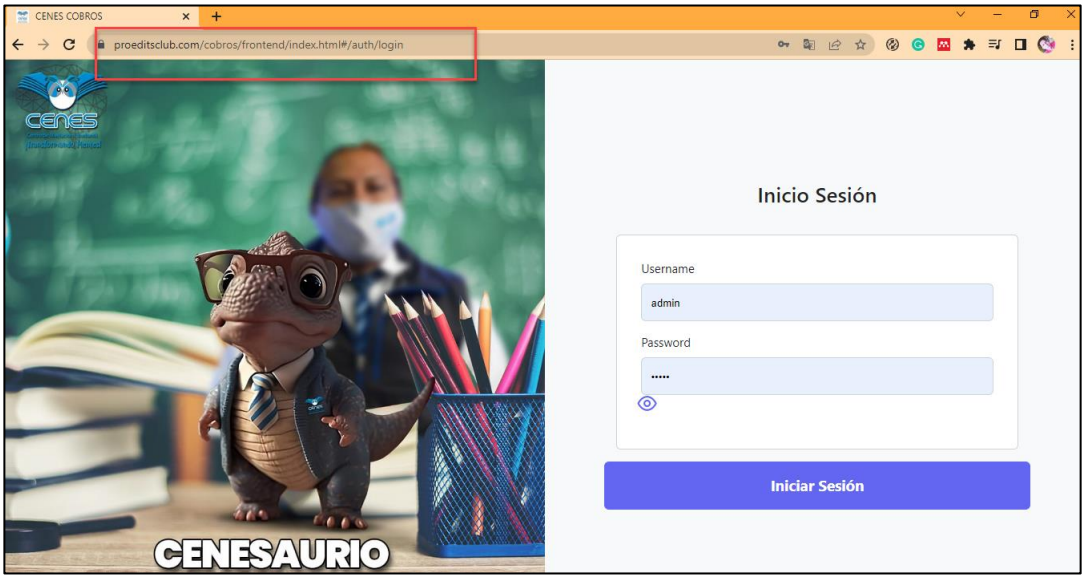


Figura 41 Funcionamiento del Frontend

5. Creación y conexión a la Base de Datos PostgreSQL.

Paso 1: En Cpanel ir a la sección de base de Datos y hacer clic en Base de datos PostgreSQL como se muestra en la Figura 42.

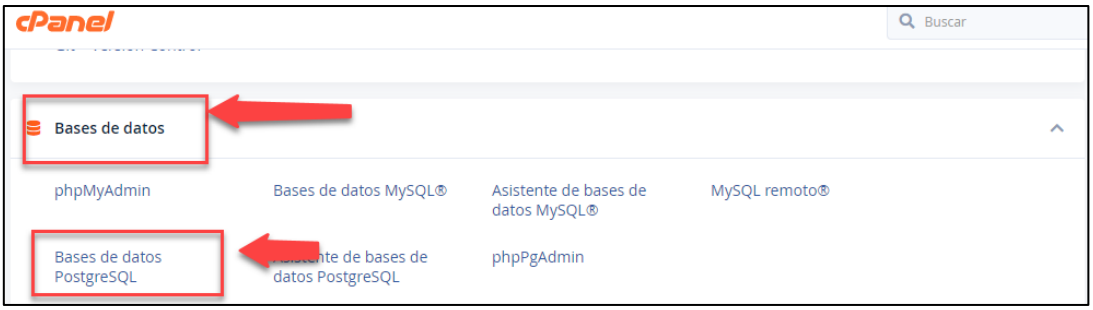


Figura 42 base de datos PostgreSQL

Paso 2: crear la base de datos como se muestra en la Figura 43.

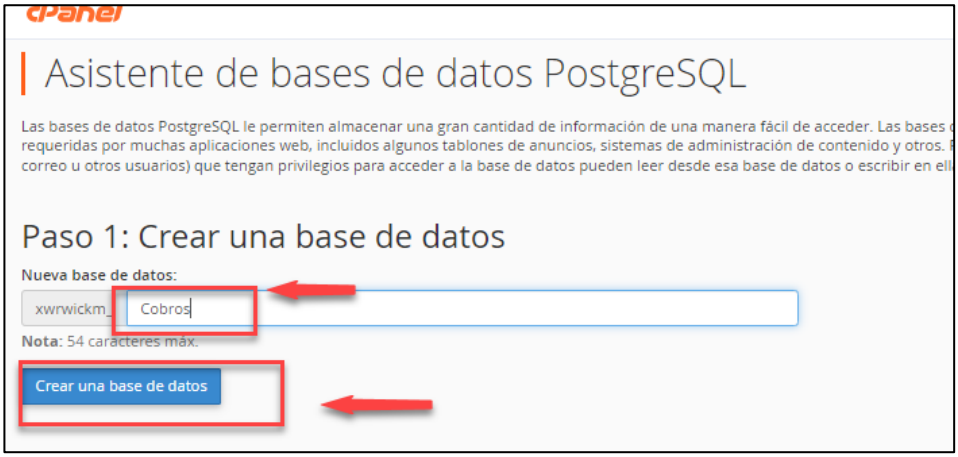


Figura 43 Crear la Base de Datos

Paso 3: Crear el Usuario ingresando el nombre de usuario y contraseñas como se muestra en la Figura 44.

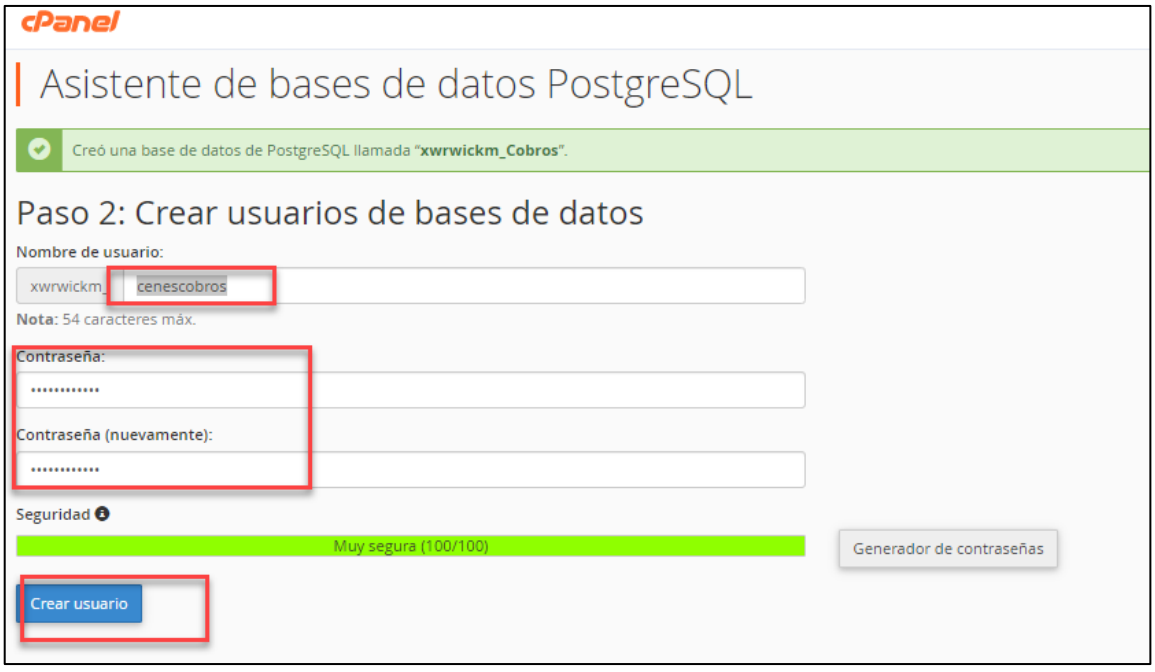


Figura 44 Crear en usuario de la Base de Datos

Paso 4: Acceder a la sección de Archivos y hacer clic en Administrador de archivos como se muestra en la Figura 45.

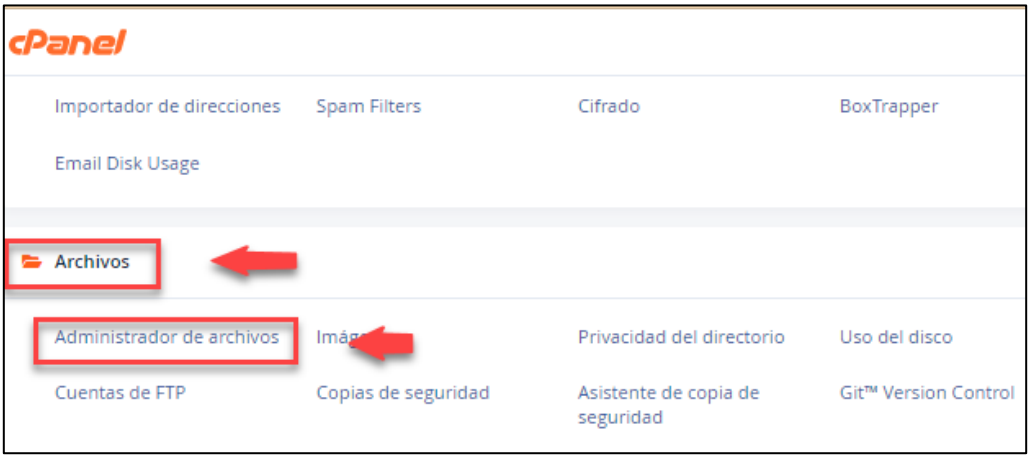


Figura 45 Sección Administrador de Archivos

Paso 5: Acceder al archivo settings.py del proyecto que se encuentra en el proyecto creado para el backend como se muestra en la Figura 46.

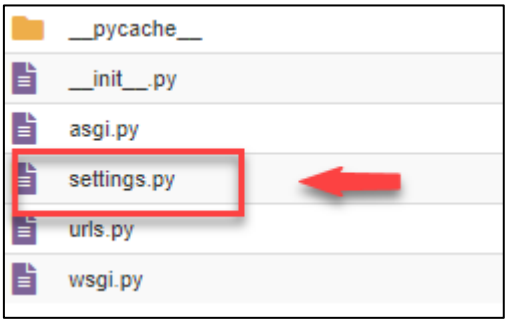


Figura 46 Archivo settings.py

Paso 6: En el apartado de DATABASES realizar la conexión de la Base de Datos como se muestra en la Figura 47.

Figura 47 Conexión a la base de datos

Paso 7: Acceder a la Terminal para realizar las migraciones mediante el comando: **python manage.py mígrate.** Como se muestra en la Figura 48.

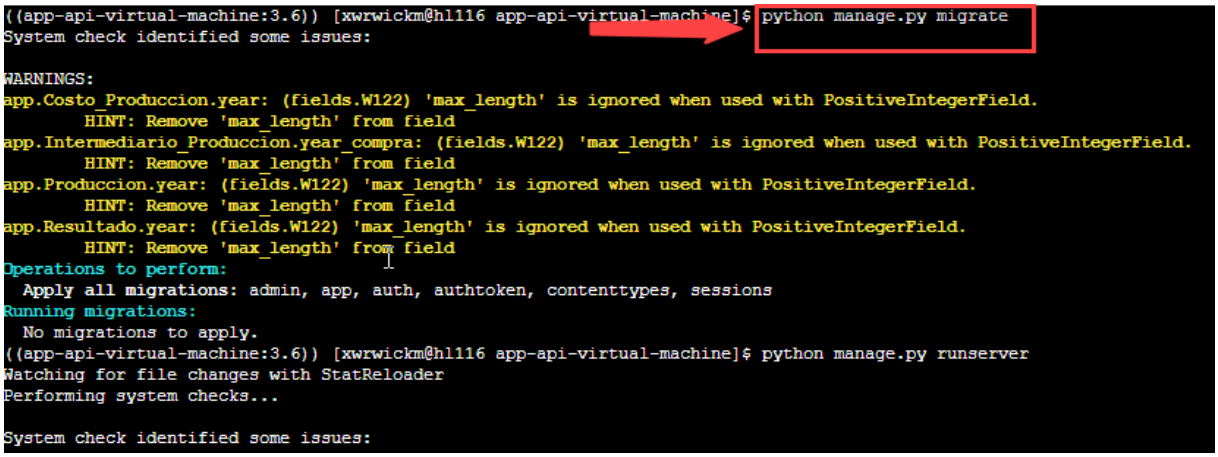


Figura 48 Migraciones en el terminal de la aplicación

Paso 8: Comprobar la base de datos creada en PostgreSQL como el gestor de base de datos phPgAdmin como se muestra en la Figura 49.

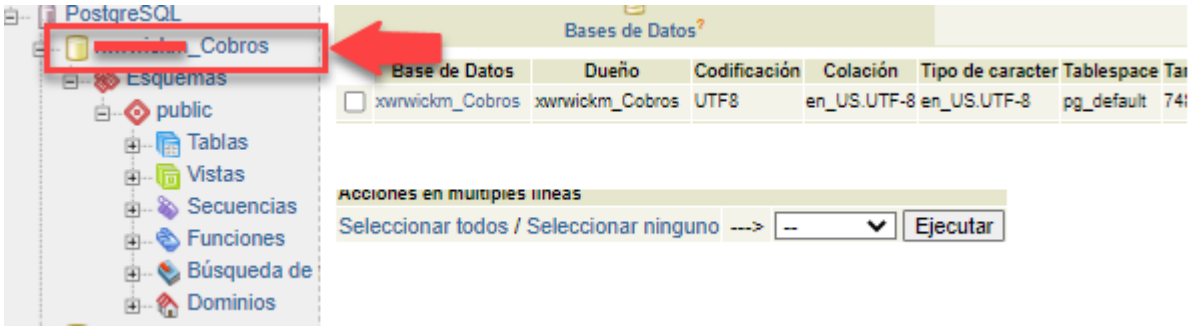


Figura 49 Comprobar la BD

Paso 9: Comprobar el funcionamiento del aplicativo web mediante e siguiente enlace https://proeditsclub.com/cobros/frontend/index.html#/auth/login como se muestra en la Figura 50.

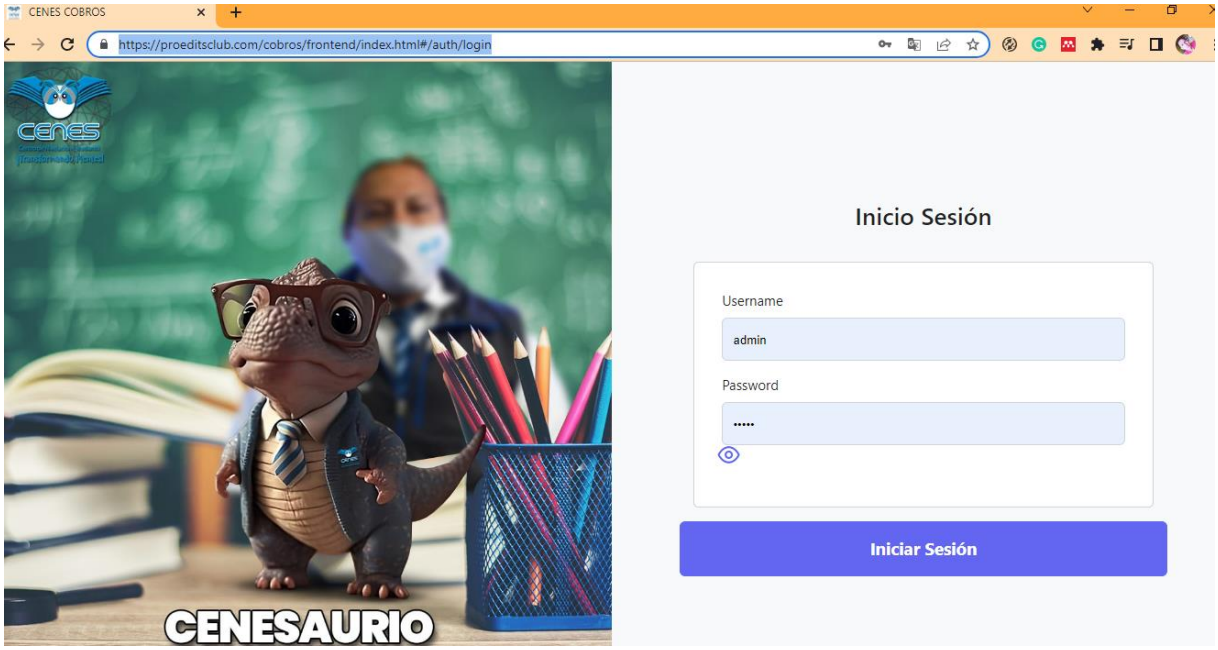


Figura 50 Vista principal del aplicativo web en producción

Anexo 11. Certificado de traducción del resumen

Loja, 24 de febrero de 2023

Lic. Marlon Armijos Ramírez Mgs.

DOCENTE PEDAGOGIA DE LOS IDIOMAS NACIONALES Y EXTRANJEROS- UNL

CERTIFICA:

Que el documento aquí compuesto es fiel traducción del idioma español al idioma inglés del resumen del Trabajo de Titulación: Desarrollo de aplicativo web para la gestión de cobro de mensualidades a estudiantes del preuniversitario "CENES". Autoria de Carla Isabel Troya Capa con CI: 1750436592, de la carrera de Ingeniería en Sistemas de la Universidad Nacional de Loja.

Lo certifico en honor a la verdad y autorizo a la interesada hacer uso del presente en lo que a sus intereses convenga.

Firmado electrónicamente por:
MARLON RICHARD
ARMIJOS RAMIREZ

MARLON ARMIJOS RAMÍREZ
DOCENTE DE LA CARRERA PINE-UNL
1031-12-1131340
1031-2017-1905329

Educamos para Transformar