



Aplicación de Soporte de evaluaciones

Diseño Industrial

Alumnos:

- Lucas Sandoval
- Tomás Toloza
- Adrian Yaniri

Profesores:

- Carlos Lombardi
- Cristian Schiffino

Materia:

- Práctica profesional supervisada

Contenido

Contenido	2
Objetivo	3
Alcances del proyecto	4
Requerimientos Funcionales	5
Módulo Administración	5
Módulo Estadístico	6
Módulo Dashboard de Actividades	7
Requerimientos no funcionales	8
Diagramas de casos de usos	10
Diagramas de arquitecturas	11
Tecnologías utilizadas	12
Frontend	12
Backend	14
Despliegue	15
Otras tecnologías	15
Aspectos desafiantes	17
Metodología de trabajo	20
Ejemplo de ficha de fin de sprint	23
Tablas de la Base de datos	25
Diseños en Figma Vs Diseños de pantallas	27
Casos de prueba	33

Objetivo

El objetivo del proyecto es poder diseñar una aplicación web que sea de ayuda para administrar las evaluaciones de los trabajos prácticos realizados por alumnos de la carrera de Diseño Industrial, que se dicta en la Universidad Nacional de Hurlingham.

La carrera cuenta con una plataforma actualmente en uso, pero a lo largo de los años, la matrícula de la carrera fue creciendo y esto hizo que la aplicación actual necesite nuevos requerimientos y funcionalidades.

Nuestro proyecto dará solución a algunas de las nuevas funcionalidades y así como también establecerá una base para que se pueda extender posteriormente.

Alcances del proyecto

La aplicación actualmente cuenta con dos tipos de roles, un rol *JTP* (Jefe de trabajos prácticos) y un rol *Estudiante*, uno de los alcances de nuestro proyecto es crear un nuevo rol *Admin* que tenga los permisos de accesos a la aplicación para poder realizar modificaciones en la misma, la de poder realizar modificaciones a los nuevos usuarios *JTP*, *Admin* y asignar los permisos a cada uno de ellos.

Además de la creación del nuevo rol, la aplicación cuenta con dos nuevos módulos, un módulo *Estadístico* y un módulo *Dashboard de Actividades*.

El módulo Estadístico tiene como objetivo realizar estadísticas relacionadas con los distintos trabajos prácticos que realizan los alumnos de la carrera.

El módulo *Dashboard de Actividades*, presentará una serie de gráficos donde se representará la información generada por los distintos tipos de datos que se van cargando en la aplicación y así como también las distintas estadísticas obtenidas en el módulo *Estadístico*

Requerimientos Funcionales

Una vez realizado el relevamiento inicial, se obtuvieron los siguientes requerimientos funcionales:

Módulo Administración

- Alta, modificación y baja de usuarios *Admin*
 - Objetivo: Permitir dar de alta, baja y modificar a un usuario *Admin*
 - Entrada: Datos del usuario *Admin*. Nombre, apellido, email y contraseña.
 - Proceso: Validar que los datos ingresados (nombre, apellido) tenga como mínimo 3 caracteres para el alta del usuario. Para eliminar o modificar un usuario se debe comprobar primero que el usuario esté registrado en la base de datos.
 - Salida: Usuario *Admin* agregado o modificado o borrado en la base de datos.

- Alta, modificación y baja de usuarios *JTP*
 - Objetivo: Permitir dar de alta, baja y modificar a un usuario *JTP*
 - Entrada: Datos del usuario *JTP*. Nombre, apellido, email y materia.
 - Proceso: Validar que los datos ingresados (nombre, apellido) tenga como mínimo 3 caracteres para el alta del usuario. Para eliminar o modificar un usuario se debe comprobar primero que el usuario esté registrado en la base de datos.
 - Salida: Usuario *JTP* agregado o modificado o borrado en la base de datos.

- Listar los usuarios *Admin*
 - Objetivo: Permitir mostrar por pantalla el listado de todos los administradores del sistema.
 - Entrada: Tipo de usuario se desea mostrar
 - Proceso:
 - Se obtiene la información de los usuarios desde la base de datos.
 - Salida: Listado con los datos de los usuarios.

- Listar los usuarios *JTP*
 - Objetivo: Permitir mostrar por pantalla el listado de todos los *JTP* del sistema.
 - Entrada: Tipo de usuario se desea mostrar.
 - Proceso:
 - Se obtiene la información de los usuarios desde la base de datos.

- Salida: Listado con los datos de los usuarios.
- Listar los usuarios *Estudiantes*
 - Objetivo: Permitir mostrar por pantalla el listado de todos los estudiantes del sistema.
 - Entrada: Tipo de usuario se desea mostrar.
 - Proceso:
 - Se obtiene la información de los usuarios desde la base de datos.
 - Salida: Listado con los datos de los usuarios.
- Listar los trabajos prácticos
 - Objetivo: Permitir mostrar por pantalla el listado de todos los trabajos prácticos del sistema.
 - Entrada: Curso y JTP que se desea visualizar
 - Proceso:
 - Se obtiene la información de los trabajos desde la base de datos.
 - Salida: Listado con los datos de los trabajos prácticos.
- Restablecer contraseñas
 - Objetivo: Permitir restablecer contraseñas para otros usuarios del sistema.
 - Entrada: Usuario a modificar y nueva contraseña
 - Proceso:
 - Se obtiene la información del usuario seleccionado y se activa el proceso de restablecimiento de contraseña
 - Salida: Email con URL del restablecimiento de la contraseña

Módulo Estadístico

- Estadísticas de aprobación de los trabajos prácticos
 - Objetivo: Obtener la estadística de todos los *Trabajos Prácticos*.
 - Entrada: Todos los *Trabajos Prácticos* realizados
 - Proceso: Obtener un listado de los *Trabajos Prácticos* aprobados (notas mayores o iguales a 4)
 - Salida: Lista con los *Trabajos Prácticos* aprobados
- Búsqueda por criterio de texto
 - Objetivo: Realizar una búsqueda en el sistema
 - Entrada: Cadena de caracteres
 - Proceso: Se filtran los elementos de los listados que cumplan la criteria
 - Salida: Listado de elementos filtrados

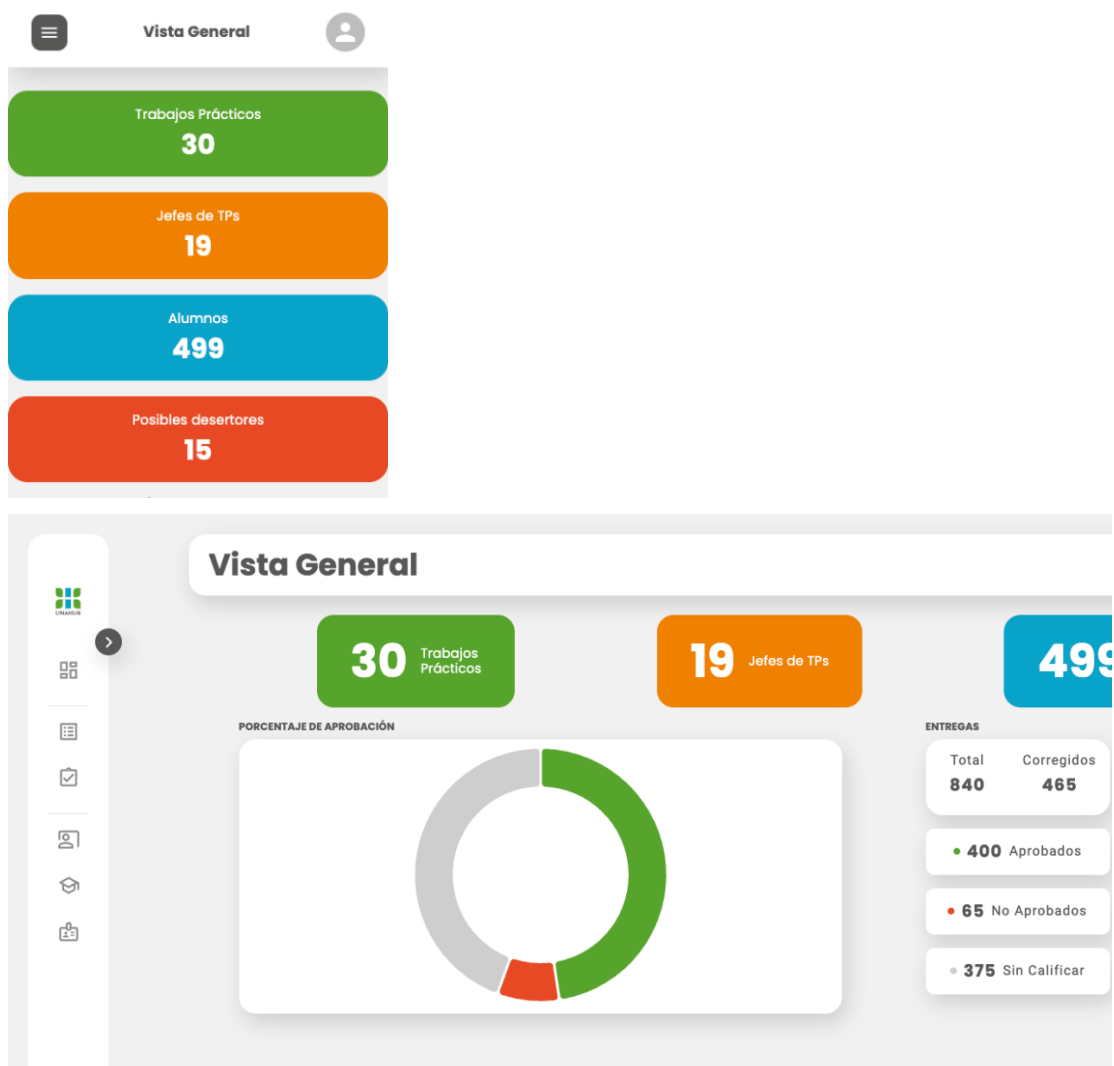
Módulo Dashboard de Actividades

- **Mostrar cantidad de *estudiantes activos***
 - Objetivo: Permitir mostrar en pantalla la cantidad de *usuarios activos*
 - Entrada: Lista de todos los usuarios
 - Proceso: Validar los estudiantes que se encuentran en el curso
 - Salida: Gráfico con la cantidad de *usuarios activos*
- **Mostrar cantidad de *TPs***
 - Objetivo: Mostrar en pantalla la cantidad de *TPs*
 - Entrada: Lista con todos los *TPs* cargados en la base de datos.
 - Proceso: Sumar todos los *TPs*
 - Salida: Gráfico con la cantidad de *TPs*
- **Mostrar porcentaje de aprobación**
 - Objetivo: Obtener el porcentaje de los *TPs* aprobados y mostrar en un gráfico
 - Entrada: Lista con todos los *TPs* cargados en la base de datos
 - Proceso: Seleccionar solo los *TPs* aprobados (notas mayores o iguales a 4), y obtener el promedio sobre el total de los *TPs*
 - Salida: Gráfico con el porcentaje de *TPs* aprobados
- **Mostrar notas por *TPs***
 - Objetivo: Obtener las notas de los *TPs*
 - Entrada: Lista de los *TPs* cargados en la base de datos
 - Proceso: Seleccionar un *TP* y obtener sus notas
 - Salida: Notas del *TP* seleccionado

Requerimientos no funcionales

- *Interfaz de usuario:* La interfaz de usuario debe ser sencilla y clara. Contar con un menú o barra lateral para poder acceder rápidamente a cualquier sección de la aplicación. Debe ser **RESPONSIVE**.

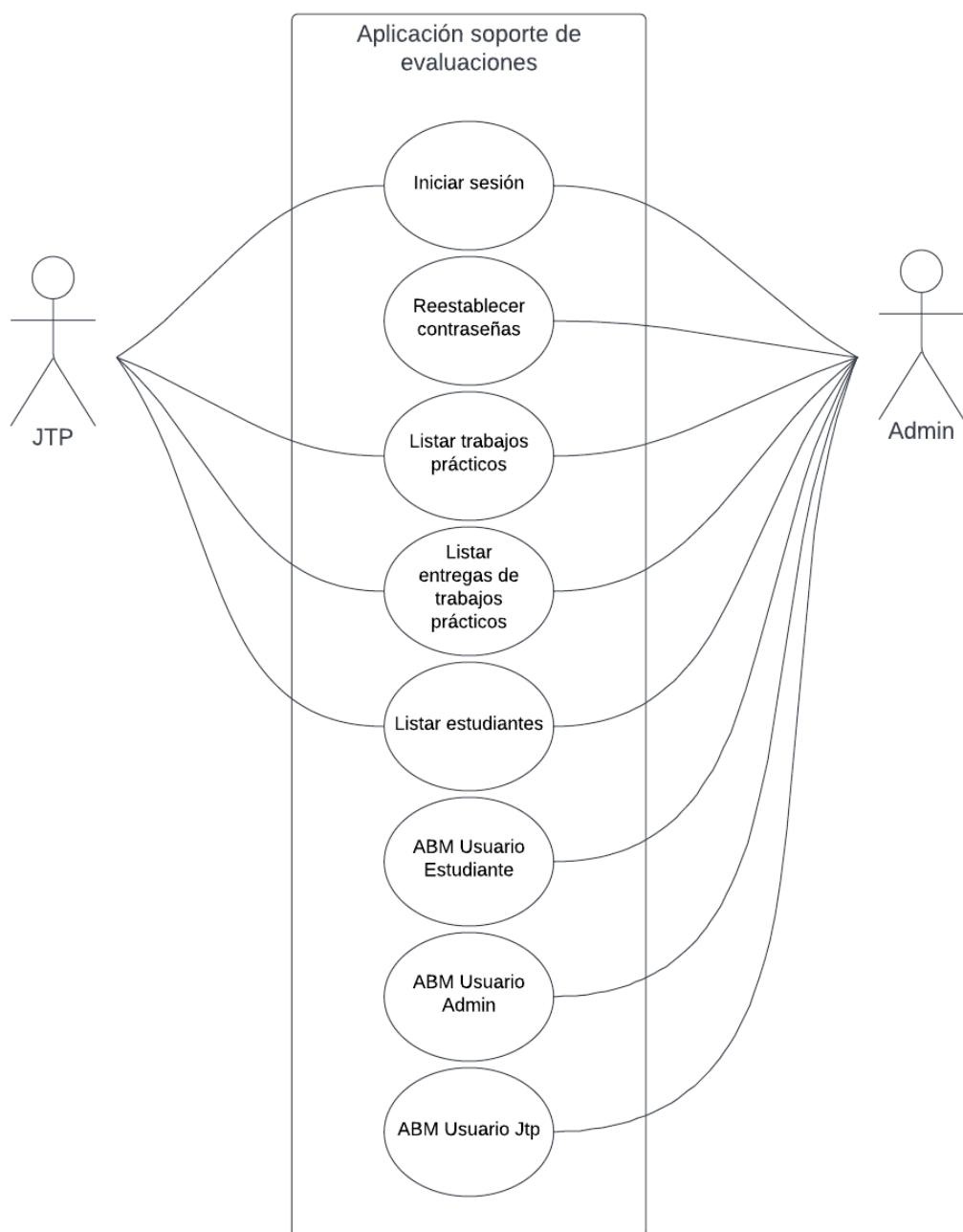
Ejemplo de pantallas Responsive



- *Portabilidad:* La aplicación debe poder funcionar en navegadores web más populares, Google Chrome, Firefox, Microsoft Edge.
- *Arquitectura:* La aplicación debe poder ser fácilmente escalable, poder agregar nuevas funcionalidades o módulos de forma rápida y sencilla. Tanto el motor de bases de datos como la aplicación facilitan este requerimiento ya que es una de las ventajas de los lenguajes y frameworks utilizados.

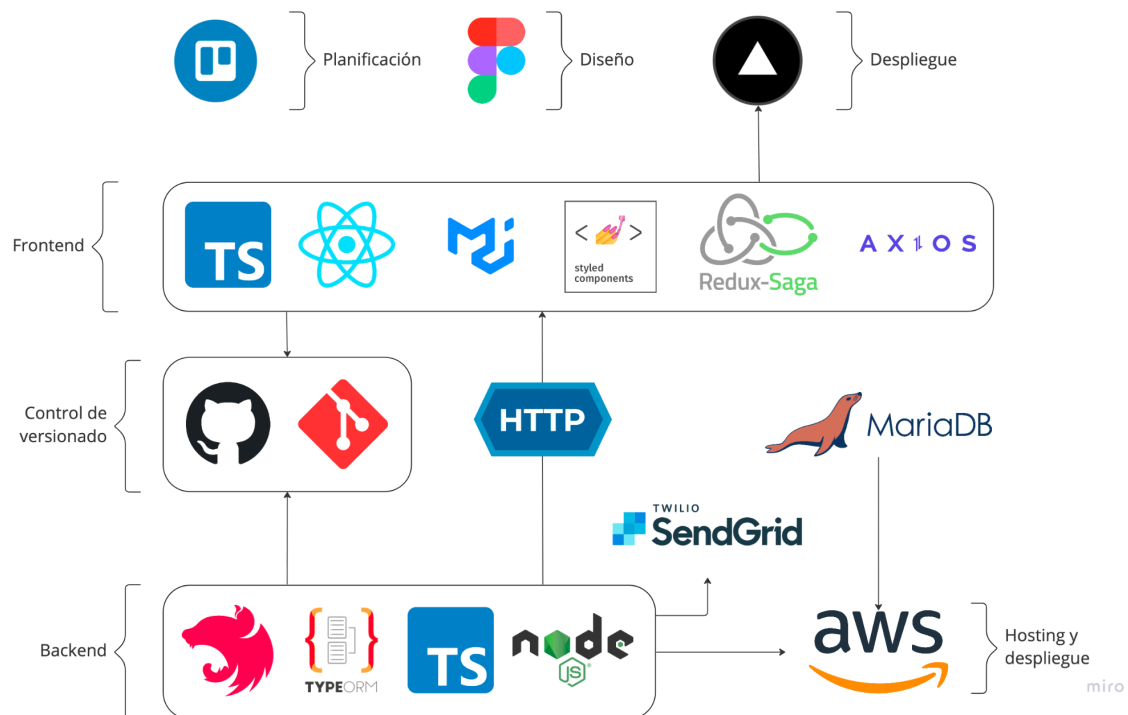
- *Disponibilidad:* Deber estar disponible para los usuarios 24hs por 7 días. Para garantizar esto la aplicación estará alojada en servidores cloud.
- *Seguridad de contraseñas:* Las contraseñas del usuario administrador en la aplicación deben estar encriptadas, para mayor seguridad.
- *Acceso al sistema:* Se realizará mediante *Login* con un nombre de usuario y contraseña. Para la validación del login entre el Frontend y el Backend se realizará mediante un Token con una duración finita pre establecida de 7 días.
- *Despliegue de la aplicación:* El frontend y el backend de la aplicación estarán alojados en servidores cloud, esto facilita a que el stakeholder no tenga que contar con servidores propios dedicados

Diagramas de casos de usos



Diagramas de arquitecturas

A continuación se muestra un diagrama donde se categorizan las distintas herramientas y tecnologías utilizadas para el proyecto. Estas se dividen en las siguientes categorías: Planificación, diseño, despliegue del frontend, frontend, control de versionado, backend, hosting y despliegue del backend



Tecnologías utilizadas

Para el desarrollo de la aplicación del frontend se utilizaron las siguientes tecnologías:

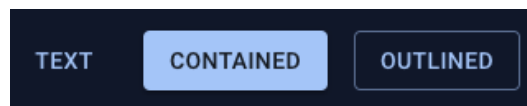
Frontend

- El lenguaje de programación se utilizó **Typescript**. **Typescript** es un superconjunto de *javascript*. Es un lenguaje de tipo estático, permite detectar los errores antes de la compilación. Se optó por este lenguaje porque permite tener un código más limpio y ordenado, y más fácil para detectar los errores de código.
- **Reactjs** se utilizó como framework para construcción de interfaces de usuarios.
Las ventajas de usar **Reactjs**, es que al estar basado en componentes facilita la creación de las interfaces de forma más sencilla y a poder manejar los estados de las mismas.
Este framework permite la reutilización de los componentes creados, así evitando repetición de código. Esto permite una mayor fluidez y un ahorro significativo en la creación de componentes
- **Material UI (MUI)** es una biblioteca de componentes de *ReactJs*. Entre las ventajas más destacadas de esta librería, permite la reutilización de componentes ya creados. Esto implica un ahorro de tiempo y código dado que su documentación provee ejemplos y una gran comunidad, así como también se obtuvo una mejor organización de código.
- **styled-components** es una librería de CSS, se puede utilizar con cualquier otro framework de *Reactjs*. La utilización en proyecto fue para darle estilos más personalizados a algunos componentes específicos. Esto permitió la organización y reutilización de componentes y estilos.

Componentes personalizados usando styled-components:



Componentes TextField y Button default de Material UI



- **Axios** es una biblioteca del cliente *HTTP*, permite realizar solicitudes al Backend de una forma sencilla y rápida. Axios tiene un buen manejo de los errores y tiene funciones que coinciden con cualquier método HTTP.
- **Redux** es una herramienta para gestionar los estados de la aplicación. Permite almacenar toda la información de los estados de los componentes en una única fuente de verdad (store). Los estados de los componentes dentro del store permanecen inmutables, y son globales permitiendo el acceso a los mismos desde cualquier componente de la aplicación.
- **Vite Js** es una herramienta de compilación para crear aplicaciones de frontend. Es agnóstica a cualquier framework. La ventaja que ofrece es una optimización en la compilación de la aplicación.

Backend

- El lenguaje de programación que se utilizó en el backend, también es **Typescript**. La elección del mismo se realizó por las ventajas mencionadas en el Frontend.
- **Nodejs** es un entorno en tiempo de ejecución de Javascript. Este permite ejecutar código de Javascript-Typescript del lado del servidor. Una de las ventajas por la cual se eligió Nodejs, es por que en hoy día es el estándar en el mercado, y cuenta con muchos foros de ayuda en los cuales se pueden consultar.
- **Nestjs** es un framework que corre sobre Nodejs para la creación de aplicaciones del lado del servidor, está construido totalmente en Typescript. Está orientado a la programación con objetos y a la programación funcional. Esta proporciona una arquitectura de aplicaciones lista para usar que permite a los desarrolladores y equipos crear aplicaciones altamente testeables, escalables, lo más desacopladas posible y fáciles de mantener.
- **TypeOrm** es un Object-Relational Mapper, o un ORM, es decir una librería utilizada para hacer de *punto* entre la base de datos relacional y el paradigma de objetos, esto permite manipular y almacenar las entidades en un formato correcto. Se eligió TypeOrm por su fácil integración con Nestjs.
- **JWT (Json Web Token)** es una herramienta que permite generar tokens de acceso. Se utilizó para permitir autenticar a los usuarios de la aplicación en el backend, cuando se inicia sesión se hace una llamada para obtener un token que luego va a ser utilizado para el resto de consultas.
- **bcrypt** es una herramienta para el encriptado de las contraseñas. Se utilizó para encriptar las contraseñas del usuario administrador solamente. Las contraseñas de los otros usuarios de la aplicación no se encuentran encriptadas, dado que encriptar las contraseñas de las entidades que son compartidas entre ambas aplicaciones, rompería compatibilidad.
- **Twilio SendGrid** es un servicio cloud para el envío de correos electrónicos, este servicio se utiliza para el restablecimiento de contraseñas de la aplicación, su fácil integración y su servicio de prueba ayudó a la implementación.

Despliegue

Para el despliegue de la aplicación se utilizó las siguientes tecnologías:

- **VERCEL** es una plataforma gratuita que permite desplegar el frontend de forma sencilla y rápida.
- **AWS** es una plataforma de *Amazon* que ofrece una capa gratuita con una instancia *EC2* para el despliegue de aplicaciones. Se utilizó para el despliegue del backend
- **Docker** es una herramienta que permite empaquetar las aplicaciones y sus dependencias en un contenedor virtual liviano. Docker permite abstraerse del sistema operativo en el cual se desarrolla la aplicación, esto hace que las aplicaciones sean más portables, y que se puedan ejecutar independientemente del sistema operativo.

Otras tecnologías

- Para el diseño de pantallas se utilizó **Figma**, es una herramienta de prototipado de diseños fácil de usar. La misma permite la interacción entre el usuario y los desarrolladores para definir la visual de la aplicación previo al desarrollo.
- La base de datos de producción fue clonada para evitar poner en riesgo la integridad de la misma. Se utilizó el servicio **RDS** de **AWS**, como plataforma de servicios en la nube, para el almacenamiento de la base de datos y para que todos los integrantes del proyecto tuvieran acceso a la misma información.
- El motor de base de datos utilizado fue **MariaDB** por ser el mismo motor que utiliza la base de datos en producción.

Las ventajas principales de usar MariaDB son:

- Es open source
- Medidas de seguridad más estrictas:
 - Verificación de contraseñas
 - Roles de usuarios
 - Cifrados de la base de datos
 - Rendimiento más rápido y más eficiente. Optimiza las consultas teniendo en cuenta las tablas independientes del motor.
 - Es compatible con otros motores de bases de datos, como por ejemplo MySQL

Al utilizar el mismo motor de base de datos, tanto en producción como en desarrollo permite utilizar una única sintaxis en las consultas, la cual reduce la posibilidad de errores al escribir las consultas que se van a realizar a la base de datos.

- **Trello** es una herramienta para gestionar los proyectos. Se la eligió por ser una herramienta recomendada por los profesores de la materia. Tiene un excelente flujo de trabajo, y una personalización según las necesidades.
- Para el control versiones de código y el almacenamiento se utilizó **GIT** y **GitHub** respectivamente. También se utilizó **GitHub Actions** para crear una integración continua de los repositorios, esto permite que todos los miembros del equipo tengan el código actualizado con sus respectivas dependencias

Aspectos desafiantes

- La base de datos al no estar normalizada aumentó la dificultad para obtener y cruzar valores en el módulo de estadísticas.
- Columnas de la base de datos ambiguas, por ejemplo, un trabajo práctico en lugar de tener una relación con otra tabla con las filas de estudiantes, tenía un varchar con los id separados por coma. Para resolver esto, se tuvo que implementar una búsqueda por id separando los valores de la columna, lo cual es intensivo y reduce el rendimiento del backend.
- La curva de aprendizaje fue desafiante al utilizar Typescript y NestJS, las primeras semanas se hizo hincapié en el aprendizaje que fue una inversión de tiempo que permitió un mejor manejo del lenguaje y del framework.
- La dificultad del **restablecimiento de contraseña** se presentó al intentar reutilizar la sección para ingresar la nueva contraseña de un usuario, en la primera versión se intentaba actualizar los usuarios desde el **PasswordResetService**, esto generaba dependencias circulares por lo que se tuvo que rediseñar los servicios. En la segunda versión solamente se utiliza el **Password Reset Service** para consultar si un restablecimiento es válido y para crear un nuevo restablecimiento, abstrayendo así sus responsabilidades.

Diagrama de flujo de la primera versión

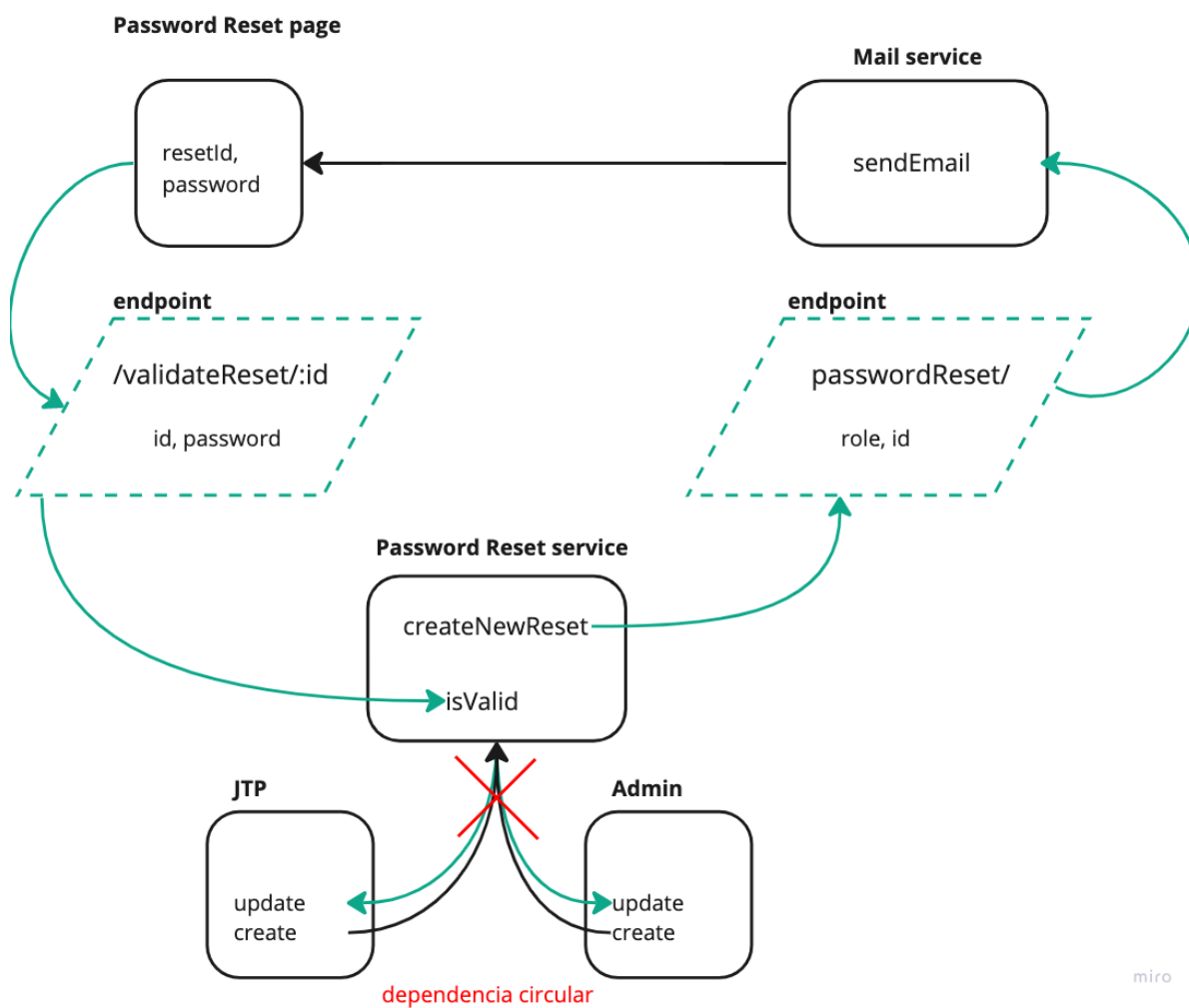
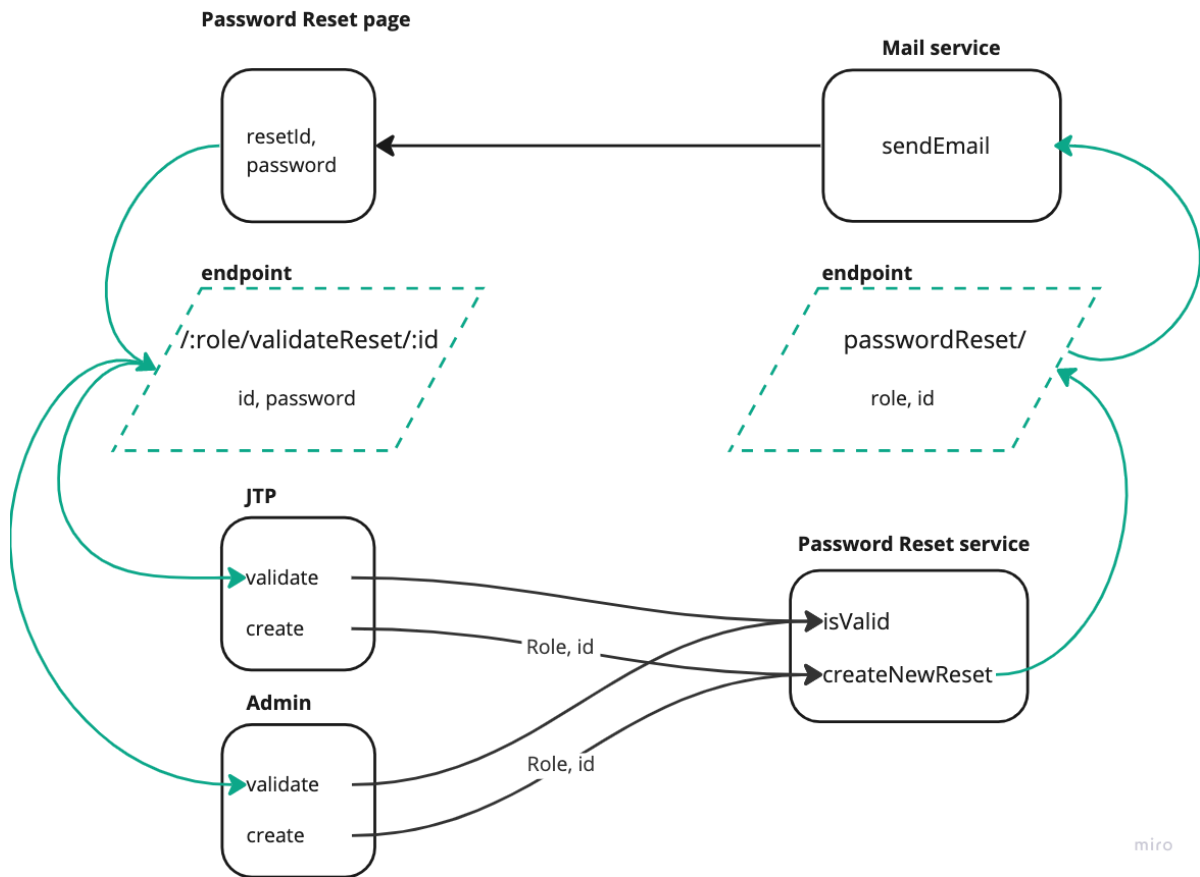


Diagrama de flujo de la segunda versión



Metodología de trabajo

Para el desarrollo de esta aplicación se utilizaron las metodologías ágiles. En particular se utilizaron la metodología de **Scrum** y de **Kanban**.

La metodología **Scrum** se basa en una estructura de desarrollo que ayuda a las personas y equipos a entregar valor de a poco de manera colaborativa e incremental. Ésta divide los proyectos en 3 etapas: análisis, desarrollo y testing. En la etapa de desarrollo encontramos lo que se conoce como interacciones del proceso o **Sprint**, es decir, entregas regulares y parciales del producto final. Esto hace que el proyecto sea mucho más manejable y con mayor previsibilidad. Así como también permite al usuario final estar involucrado en un feedback periódicamente.

La planificación del proyecto se realizó de la siguiente manera:

- Se dividió en 10 sprints.
- Cada sprint tiene una duración de 3 semanas.
- Una reunión de planificación al inicio de cada sprint.
- Una reunión de seguimiento cada semana.
- Reunión de revisión fin de sprint.
- Elaboración de una ficha de inicio y fin de sprint.

La metodología de **Kanban** se utilizó para proveer transparencia y ayuda visual en la asignación de las tareas, para llevarla a cabo, se utilizó **Trello**.

Trello es una herramienta para gestionar proyectos. Esta permite crear tableros con tarjetas de las diferentes tareas a desarrollar y poder asignarle un desarrollador para que las realice. Para este proyecto se crearon 3 tableros:

- **Este sprint** - Para las tareas del sprint actual
- **Futuro** - Para las tareas de un sprint futuro
- **Pasado** - Para las tareas que ya fueron realizadas

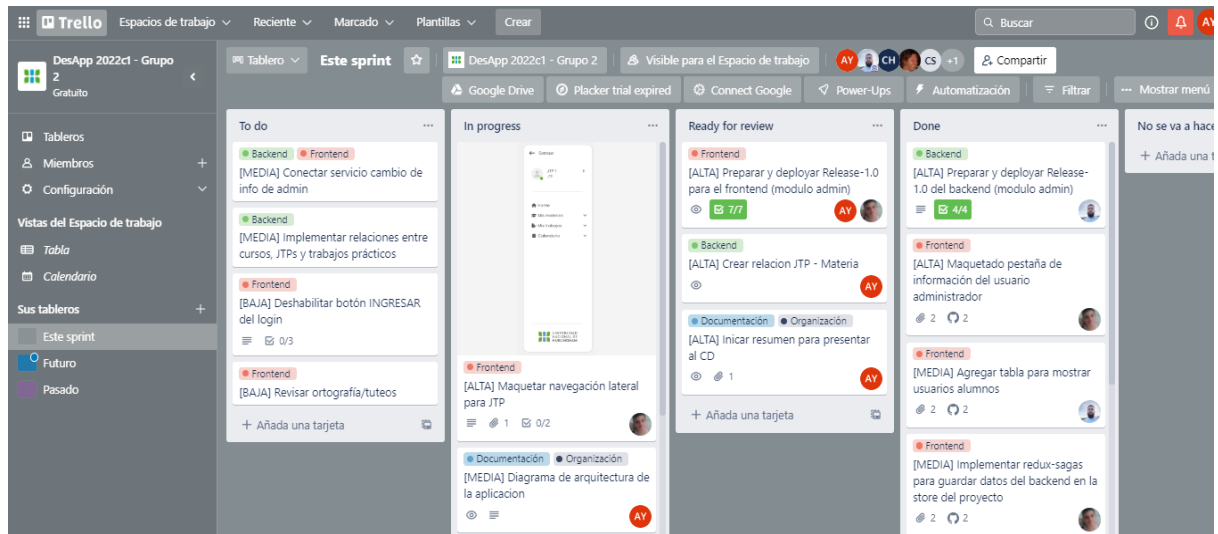
A su vez el tablero de **Este sprint** se divide en columnas donde se ubican las tareas.

- **To do** para las tareas a realizar
- **In progress** para las tarea que fueron asignadas y están en proceso de desarrollo
- **Ready for review** para las tareas que están en revisión
- **Done** las tareas que ya se realizaron
- **Próximo sprint** para tareas que posiblemente se realicen en el próximo sprint

Para una mayor organización de las tareas se decidió que a cada tarjeta se le coloque una etiqueta con niveles de prioridad. Estos niveles son:

- **Alta** para tareas de mayor prioridad
- **Media** para las tareas con una prioridad media
- **Baja** para las tareas que no son prioritarias

Ejemplo de un tablero de Trello



Ejemplo de ficha de inicio de sprint

Desarrollo de aplicaciones - 1er cuatrimestre 2022

Ficha de principio de sprint

Grupo 2 - Diseño Industrial
Aplicación para soporte evaluación Diseño Industrial

Nro de sprint	Fecha de comienzo	Fecha de fin
2	28/04/2022	19/05/2022

Objetivos

- Definir estructura de las historias de usuario
- Revisar listas de Trello
- Realizar Ficha de fin del Sprint 2
- Definir el stack tecnológico a utilizar en el proyecto. (Database)
- Modelar primer diseño de la base de datos
- Crear el repositorio para el backend
- Crear el repositorio para el frontend
- Crear especificación del Modulo de administración de usuarios
- Crear especificación del modelo Trabajo Práctico
- Crear especificación del modelo Usuario
- Coordinar reunión con Germán
- Definir branch strategy
- Definir paleta de colores
- Diseñar pantalla Trabajos prácticos (módulo administración)
- Diseñar pantalla Perfil de usuario

Ejemplo de ficha de fin de sprint

Desarrollo de aplicaciones - 1er cuatrimestre 2022

Ficha de fin de sprint

Grupo 2 - Diseño Industrial

Nro de sprint	Fecha de comienzo	Fecha de fin
2	28/04/2022	19/05/2022

Objetivos que nos pusimos para este sprint (una frase para cada uno).

- Definir estructura de las historias de usuario
- Revisar listas de Trello
- Realizar Ficha de fin del Sprint 2
- Definir el stack tecnológico a utilizar en el proyecto. (Database)
- Modelar primer diseño de la base de datos
- Crear el repositorio para el backend
- Crear el repositorio para el frontend
- Crear especificación del Modulo de administración de usuarios
- Crear especificación del modelo Trabajo Práctico
- Crear especificación del modelo Usuario
- Coordinar reunión con Germán
- Definir branch strategy
- Definir paleta de colores
- Diseñar pantalla Trabajos prácticos (módulo administración)
- Diseñar pantalla Perfil de usuario

A qué llegamos, o nos acercamos bastante.

Definir paleta de colores

70% Se definió la paleta de colores. Falta definir con Germán

Diseñar pantalla Trabajos prácticos (módulo administración)

60% Se avanzó con el listado del TPs. Falta pantalla de edición/creación de TPs

Diseñar pantalla Perfil de usuario

80% Se avanzó con la pantalla del perfil de usuario. Falta revisar últimos detalles.

A qué no llegamos.

Para cada cosa a la que no se llegó, contar brevemente **por qué**, o sea, qué problemas aparecieron.

- Crear especificación del Modulo de administración de usuarios
- Crear especificación del modelo Trabajo Práctico
- Crear especificación del modelo Usuario

Se decidió no realizar las tareas debido a la absorción de ellas en la tarea del modelado del DER.

- Coordinar reunión con Germán

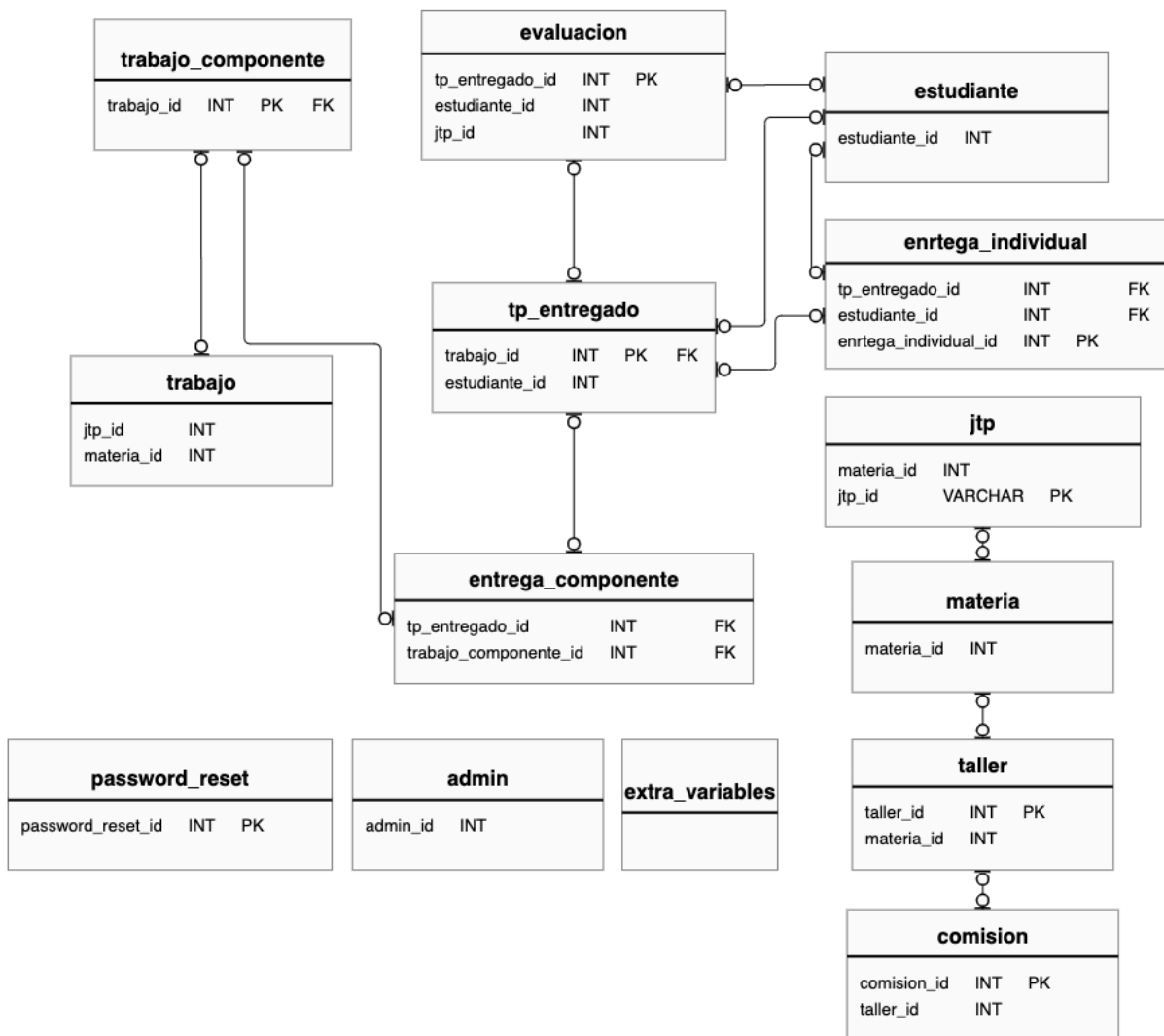
No pudimos coordinar con el usuario en este sprint, se decidió posponer la reunión para poder presentar más avances.

Tablas de la Base de datos

El DER (diagrama entidad relación) de la siguiente imagen pertenece a las tablas de la base de datos que se encuentra actualmente en producción. La base de datos de producción se clonó y se alojó en una instancia de *AWS*, esto permite tener acceso a la base de datos y poder realizar modificaciones en la base de datos de producción sin alterar la original.

Se respetó la misma arquitectura y motor de la base de datos de la plataforma actual, esto permite realizar las consultas a la base de datos con la misma sintaxis y poder utilizar un *script* con consulta que ha sido facilitado por el *stakeholder*.

A continuación se muestra un diagrama entidad relación simplificado de la base de datos:



Diseños en Figma Vs Diseños de pantallas

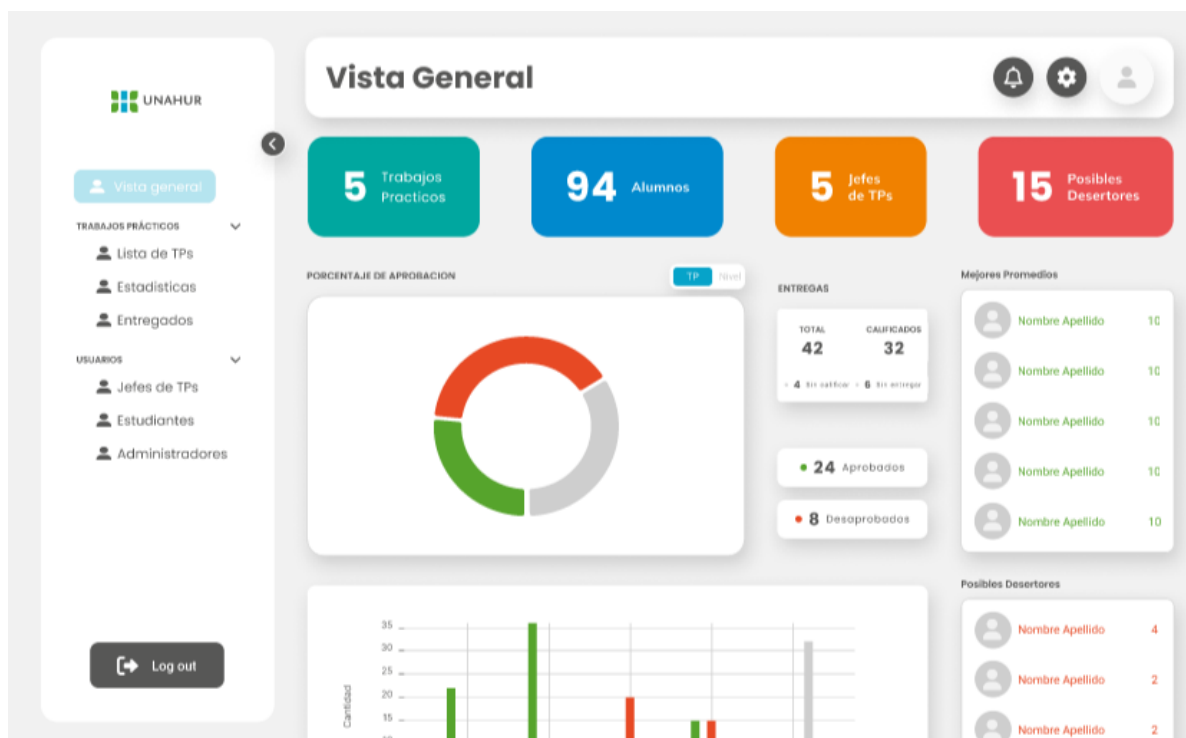
Pantalla de login - Figma

The Figma design shows a login screen with a white background and rounded corners, set against a dark gray frame. At the top is the UNAHUR logo, consisting of a 2x2 grid of squares (green, blue, green, blue) with the text 'UNAHUR' below it. The title 'Ingresá a tu cuenta' is centered below the logo. The form includes an 'Email' label, a text input field with a light blue border containing 'User@unahur.edu.ar' and an envelope icon, a 'Contraseña' label, a password input field with a light blue border containing masked characters and a lock icon, and a link '¿Olvidaste tu contraseña?' below the password field. A green 'Ingresar' button is at the bottom.

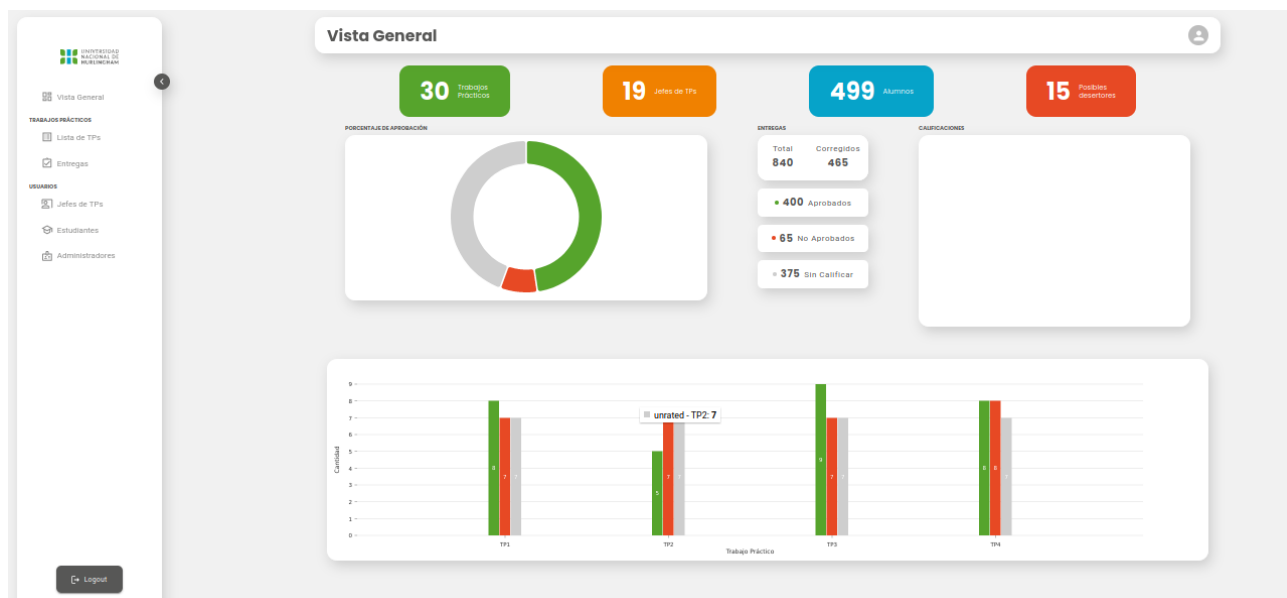
Pantalla login - Final

The final design shows a login screen with a white background and rounded corners, set against a dark gray frame. At the top is the UNAHUR logo, consisting of a 2x2 grid of squares (green, blue, green, blue) with the text 'UNAHUR' below it. The title 'Ingresá a tu cuenta' is centered below the logo. The form includes an 'EMAIL' label, a text input field with a light gray border containing 'test@unahur.com' and an envelope icon, a 'CONTRASEÑA' label, a password input field with a light gray border containing masked characters and a lock icon, and a green 'Ingresar' button at the bottom.

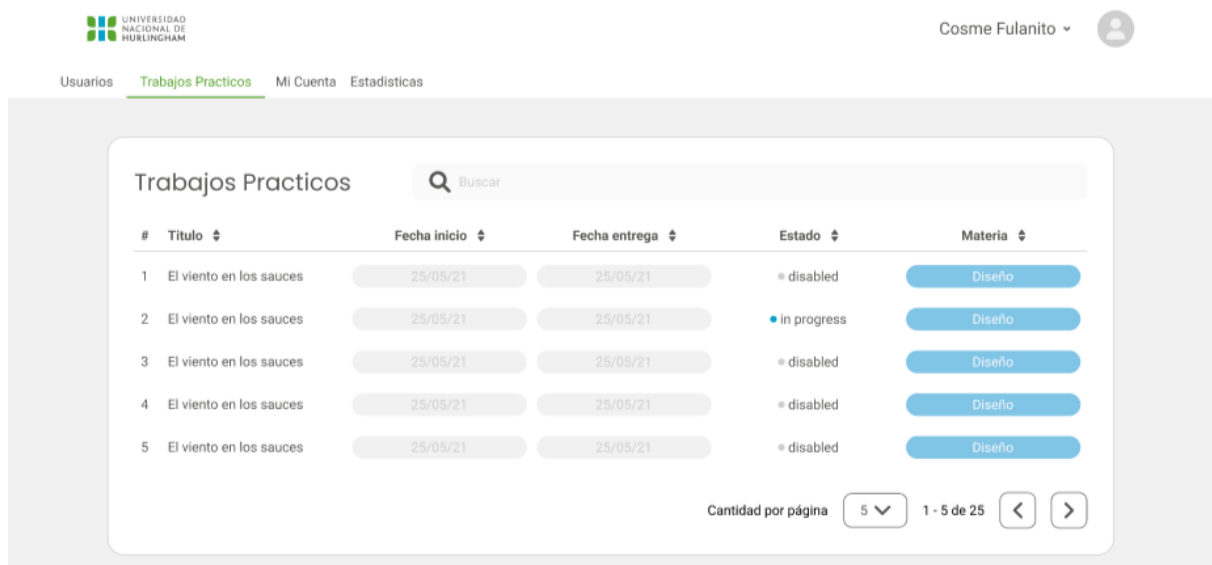
Pantalla dashboard - Figma



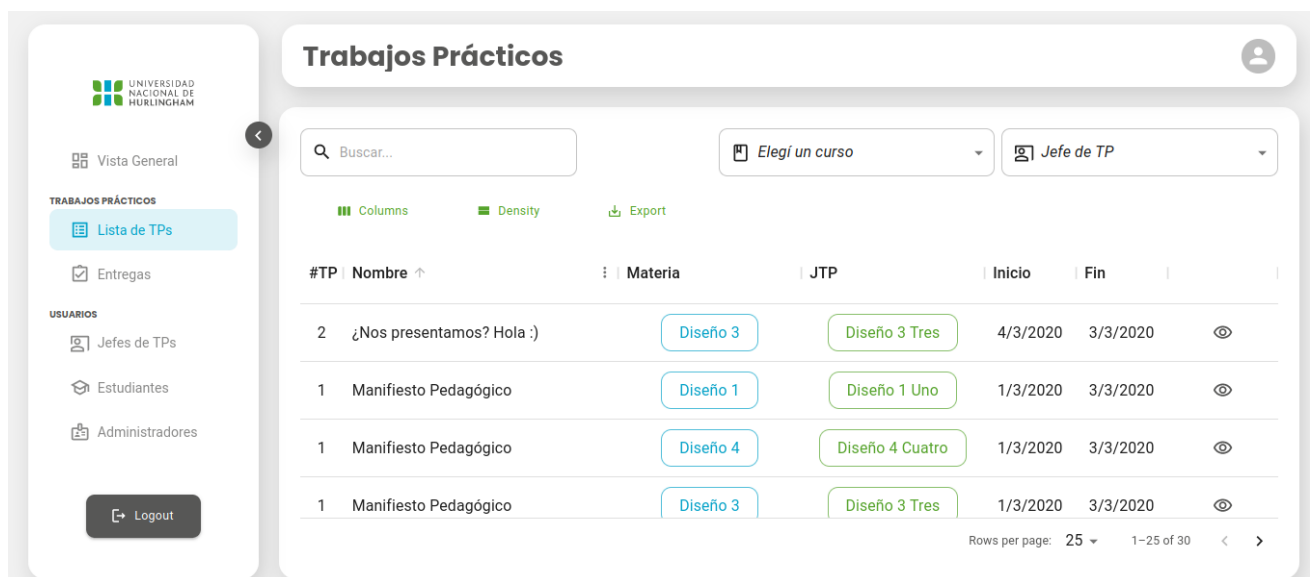
Pantalla dashboard - Final



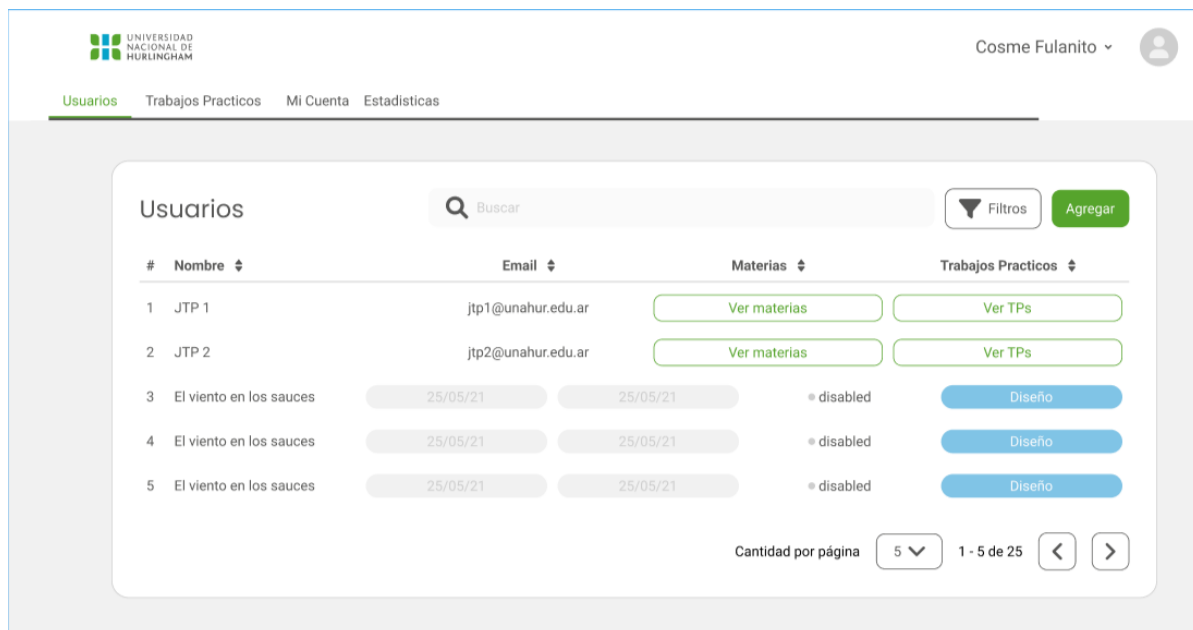
Pantalla Lista de Trabajos Prácticos - Figma



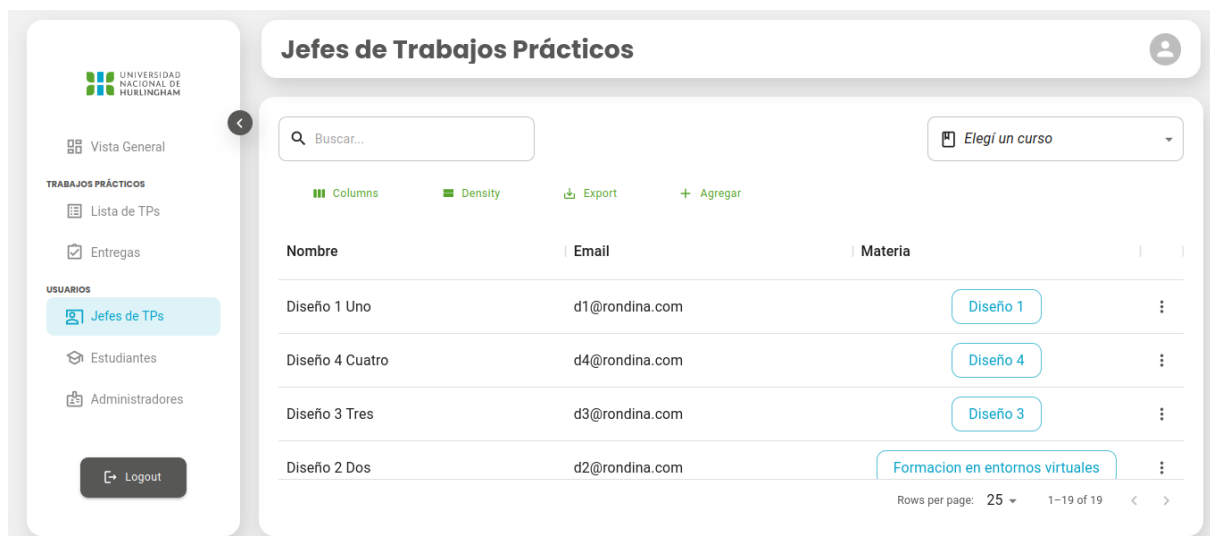
Pantalla Lista de Trabajos Prácticos - Final



Pantalla de JTP - Figma



Pantalla de JTP - Final



Modal para agregar JTP - Figma

UNIVERSIDAD NACIONAL DE HURLINGHAM

Cosme Fulanito

Usuarios Trabajos Practicos Mi Cuenta Estadísticas

← Volver

Crear nuevo usuario

Nombre/s

Apellido

Email

DNI

Rol

Materias

Modal para agregar JTP - Final

Trabajos Practicos

Elegí un c...

Density

Agregar nuevo Jefe de TP X

NOMBRE

APELLIDO

EMAIL

CURSO

tecno@rondina.com

Formacion en entornos virtuales

Diseño 3

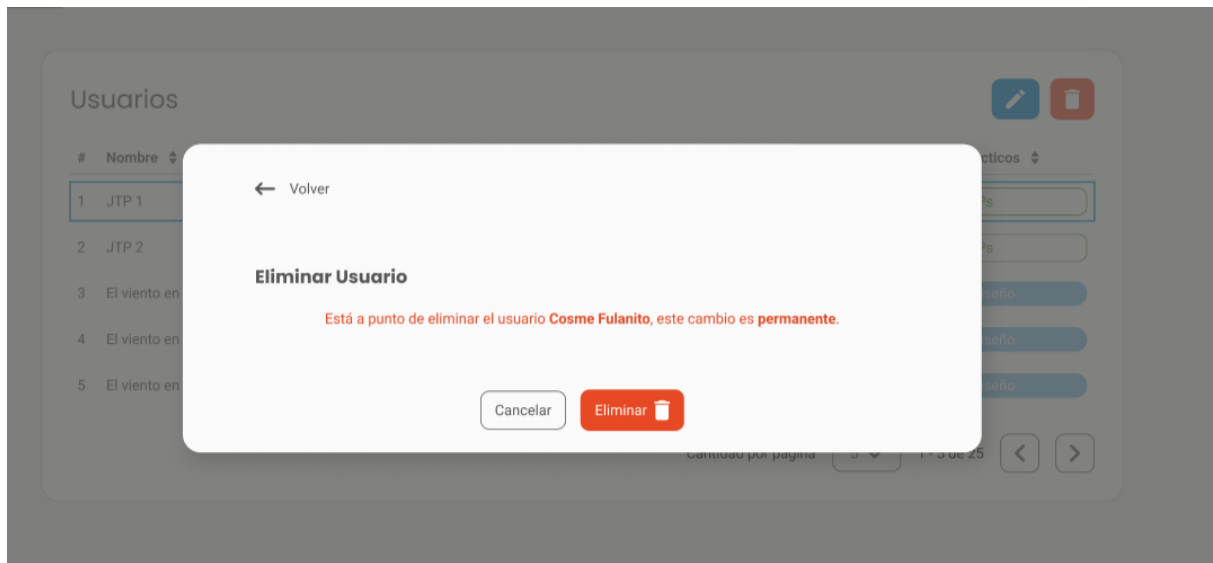
Diseño 5

Diseño 3

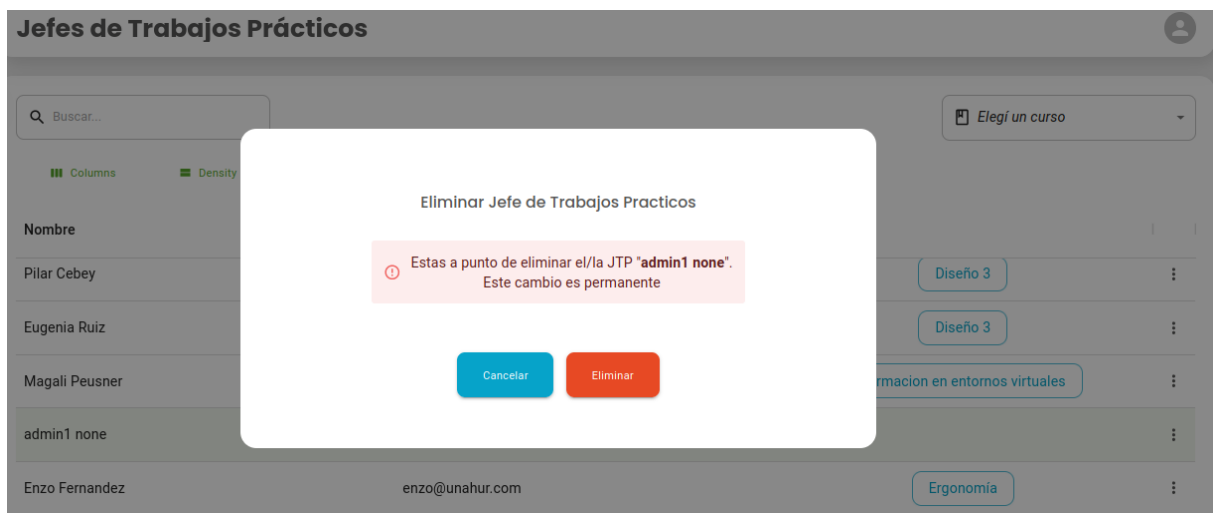
Tecno

Rows per page: 25

Modal para eliminar un usuario - Figma



Modal para eliminar un usuario - Final



Casos de prueba

Para los siguientes casos de pruebas se utilizó la metodología de pruebas unitarias.

Los test se hicieron sobre los principales requerimientos funcionales. Cada caso prueba contará con los siguientes ítems:

- Título
- Descripción
- Precondición
- Entrada o input
- Salida o output

Además se adjuntará una captura de pantalla de cada uno


Título	Iniciar sesión
Descripción	Un usuario JTP inicia sesión con su email y contraseña
Precondición	El usuario JTP debe existir en la base de datos
Entrada	Email y contraseña
Comportamiento esperado	El usuario se loguea exitosamente



UNAHUR

Ingresá a tu cuenta


EMAIL


 euge@rondina.com

CONTRASEÑA





Ingresar


Mi Cuenta



Eugenia Ruiz

 euge@rondina.com

 Administrador

 Editar

Título	Crear un Jtp
Descripción	Debe existir un usuario con el rol de Admin
Precondición	Debe existir un usuario con el rol Admin
Entrada	Nombre, apellido, email, curso del JTP
Comportamiento esperado	Se crea el usuario y se visualiza en el listado

The screenshot shows a modal window titled 'Agregar nuevo Jefe de TP' with a close button (X). The form contains four input fields: 'NOMBRE' with the value 'Enzo', 'APELLIDO' with the value 'Fernandez', 'EMAIL' with the value 'enzo@unahur.com', and 'CURSO' which is a dropdown menu currently showing 'Matemática'. A green 'Confirmar' button with a checkmark is at the bottom center. The background shows a table of 'Jefes de Trabajos Practicos' with columns for 'Nombre' and 'Email'.

The screenshot shows a table with the following data:

Nombre	Email	Materia
Magali Peusner	magali.peusner@unahur.edu.ar	Formacion en entornos virtuales
Test Test	test@unahur.com	Matemática
admin1 none	tomas.toloza@estudiantes.edu.ar	
Enzo Fernandez	enzo@unahur.com	Matemática

The table has a search bar at the top left, a dropdown for 'Elegí un curso' at the top right, and a 'Agregar' button. The footer shows 'Rows per page: 25' and '1-20 of 20'.

Título	Modificar un Jtp
Descripción	Un usuario admin crea un usuario JTP
Precondición	Debe existir un usuario con el rol de Admin
Entrada	Nombre, apellido, email o curso del JTP
Comportamiento esperado	Se modifica el usuario y se visualiza el cambio en el listado

Mi Cuenta

Test Test

test@unahur.com

Administrador

Editar

Jefes de Trabajos Prácticos

Editar Jefe de trabajos Practicos

NOMBRE: Enzo

APELLIDO: Fernandez

EMAIL: enzo@unahur.com

CURSO: Ergonomía

Confirmar


Columns Density Export + Agregar

Nombre	Email	Materia
Magali Peusner	magali.peusner@unahur.edu.ar	Formacion en entornos virtuales
Test Test	test@unahur.com	Matemática
admin1 none	tomas.toloza@estudiantes.edu.ar	
Enzo Fernandez	enzo@unahur.com	Ergonomía

1 row selected

Título	Eliminar un Jtp
Descripción	Un usuario admin elimina un usuario JTP
Precondición	Debe existir un usuario con el rol de Admin
Entrada	Jtp a eliminar
Comportamiento esperado	Se elimina el usuario y se visualiza la actualización en el listado

Mi Cuenta

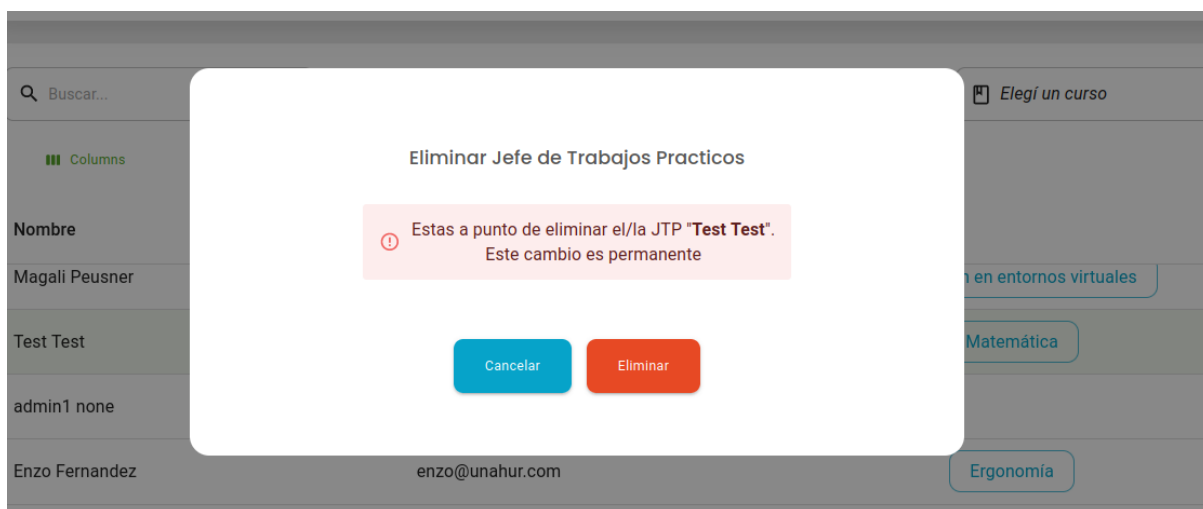


Test Test

test@unahur.com

Administrador

Editar



Jefes de Trabajos Prácticos			
<input type="text" value="Buscar..."/>	<div>Elegí un curso</div>		
Columns	Density	Export	Agregar
Nombre	Email	Materia	
Eugenia Ruiz	euge@rondina.com	Diseño 3	
Magali Peusner	magali.peusner@unahur.edu.ar	Formacion en entornos virtuales	
admin1 none	tomas.toloz@estudiantes.edu.ar		
Enzo Fernandez	enzo@unahur.com	Ergonomía	
<div>Rows per page: 25 1-19 of 19</div>			

Título	Listar trabajos prácticos
Descripción	Al elegir la sección de trabajos prácticos, se muestra el listado con información de cada uno
Precondición	Deben existir trabajos prácticos
Entrada	
Comportamiento esperado	Se listan los trabajos prácticos

Trabajos Prácticos

Buscar...

Elegí un curso

Jefe de TP

Columns

Density

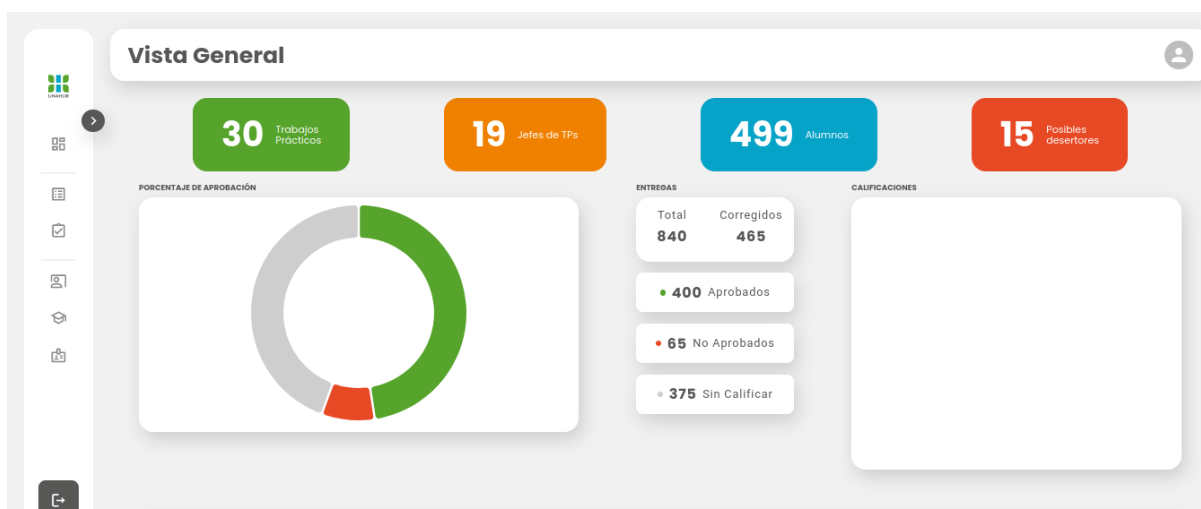
Export

#TP	Nombre	Materia	JTP	Inicio	Fin	
2	¿Nos presentamos? Hola :)	Diseño 3	Diseño 3 Tres	4/3/2020	3/3/2020	
1	Manifiesto Pedagógico	Diseño 1	Diseño 1 Uno	1/3/2020	3/3/2020	
1	Manifiesto Pedagógico	Diseño 4	Diseño 4 Cuatro	1/3/2020	3/3/2020	
1	Manifiesto Pedagógico	Diseño 3	Diseño 3 Tres	1/3/2020	3/3/2020	
2	Sentirse Bien	Diseño 3	Diseño 3 Tres	2/11/2020	3/11/2020	

Rows per page: 25

1-25 of 30

Título	Estadísticas usuario Admin
Descripción	El usuario “admin” visualizará en el dashboard la información de todos los trabajos prácticos, todos los JTP y estudiantes que están cargados en el sistema
Precondición	Debe existir un usuario con el rol de Admin
Entrada	Usuario Admin logueado
Comportamiento esperado	Se mostrar en la pantalla el total de todos los JTP, estudiantes y trabajos prácticos cargados en el sistema



Título	Estadísticas usuario JTP
Descripción	El usuario “JTP” solo visualizará en el dashboard la información de todos los trabajos prácticos donde es docente
Precondición	Debe existir un usuario con el rol de JTP
Entrada	Usuario JTP logueado
Comportamiento esperado	El dashboard solo mostrará los trabajos prácticos que el JTP fue asignado

