



UNIVERSIDAD NACIONAL
DE HURLINGHAM

PRÁCTICA PROFESIONAL SUPERVISADA
2023 - 2C

SISTEMA PARA GESTIÓN DE SOLICITUDES DE EQUIVALENCIAS

ALUMNOS

Ahumada, Emir
Galván, Mariano Ezequiel
Gutiérrez, Raúl Amadeo
Páez, Julián Ezequiel
Robledo, Javier Ángel

PROFESORES

Schiffino, Cristian
Lombardi, Carlos
Carboni, Miguel



Contenido

Antecedentes y Descripción.....	2
Situación inicial relevada.....	2
Objetivos del proyecto.....	4
Requerimientos funcionales.....	5
Requerimientos no funcionales.....	6
Diagrama de caso de uso.....	7
Plan de trabajo.....	8
Integración inicial.....	12
Integración final.....	14
Reunión con el Stakeholder.....	15
Cronograma de tareas.....	15
Casos de pruebas.....	16
Entrega y despliegue.....	23
Despliegue local:.....	23
Despliegue remoto:.....	26
Despliegue de la base de datos:.....	31
Requerimientos pendientes.....	36

Antecedentes y Descripción

El proyecto surge de la necesidad de desarrollar una plataforma, por parte de la Universidad de Hurlingham, que resuelva la complejidad de recibir, administrar y documentar las solicitudes de equivalencias para materias aprobadas en otras universidades, como una primera instancia en el trámite de este tipo de solicitud.

En cursadas anteriores de Desarrollo de Aplicaciones y Práctica Profesional Supervisada, distintos grupos de estudiantes desarrollaron esta aplicación, y realizaron diversos requerimientos que han dado lugar a distintas versiones del mismo. Aunque estas versiones aún no están implementadas, poseen distintas funcionalidades y herramientas que deben integrarse para lograr una primera versión operativa de la aplicación.

La aplicación, posibilita al director revisar y gestionar solicitudes de equivalencias, mediante la asignación de estados para aceptar, rechazar o solicitar información adicional para su aprobación, que facilitan así un diálogo interactivo y fluído con los alumnos.

En el caso de que la solicitud de la materia pedida en la equivalencia sea “aceptada” por la dirección, los alumnos podrán iniciar el trámite de manera tradicional mediante la obtención de un formulario PDF, el cual será informado desde la oficina de Equivalencias. Si la respuesta al pedido es “rechazada”, el usuario recibirá una devolución, enviada desde la aplicación a su correo electrónico registrado, en donde se le explicará la razón por la cual se llegó a ese resultado. Por último, el estado de “pendiente” significa que, hay información faltante por parte del alumno, o bien falta la revisión por parte del directivo, todo esto desde la aplicación, como se comentó para el caso anterior.

Situación inicial relevada

Se recibieron tres versiones iniciales, que por separado le dan forma al proyecto a integrar por nuestro equipo. En primera instancia, como parte del proceso de evaluación, se realizó por separado la implementación de cada versión, para ver su funcionamiento de forma individual. Un aspecto a destacar es que todas las versiones, inicialmente, se encontraban operativas en gran parte. Cada grupo, guiado por su visión y experiencia, pudo diseñar y desarrollar un sistema que aborda la administración de equivalencias de una manera efectiva. Los tres proyectos están

publicados en las siguientes direcciones de Github: [Frontend](#) y [Backend](#). Y poseen las siguientes extensiones como requerimientos:

Extensión 1 (Grupo 1)

1. CRUD de universidades, persistencia en la BD, y usar las universidades en el formulario de equivalencia.
2. Agregar la carga de un PDF para cada materia aprobada en el pedido de equivalencia.
3. Agregar validaciones en la carga de equivalencias.
4. Pop-up o cambiar de página cuando el estudiante hace click sobre “Perfil de usuario”.
5. Definir un header específico para el rol directivo.
6. Funcionalidad asociada al botón “Registrarse” en la página de login.
7. Encriptar contraseñas en la BD.

Extensión 2 (Grupo 4)

- 1.- Agregar estado en los trámites de equivalencia para rol directivo y un filtro que permite ver sólo los trámites pendientes.
- 2.- Revisar el circuito de carga y control de equivalencias.
- 3.- Pasar a un modelo en el cual un mismo trámite de equivalencia puede incluir varias materias. En este modelo, el directivo puede aprobar, dentro de un mismo trámite, algunas materias sí y otras no. Los comentarios son globales. Un trámite está cerrado cuando hay una decisión para cada una de las materias solicitadas.

Extensión 3 (Grupo 3)

- 1.- CRUD de carreras UNAHUR, persistencia en la BD, y usar en el formulario de nueva equivalencia.
- 2.- Asignar al rol directivo para que puedan ver solamente los trámites que corresponden a “sus carreras”.
- 3.- Agregar un nuevo rol “superusuario”, que pueda ver todos los trámites, y que también pueda acceder a una tabla con todos los usuarios, que permite filtrar por rol directivo o estudiante.
- 4.- Transformar el recuadro de “respuesta” en un chat asociado específicamente al trámite de equivalencia.

Objetivos del proyecto

1. Integración en un repositorio principal

Unificar las tres versiones independientes del mismo proyecto, realizar mejoras y dejar en funcionamiento el proyecto. Planificar y ejecutar refactorizaciones esenciales para garantizar una ejecución de la aplicación en un ambiente de producción, facilitar la acumulación de cambios mediante la documentación de cada proceso o sprint que ha sido completado con metodologías ágiles que se han utilizado para la gestión y organización del trabajo.

2. Despliegue en la nube

Desplegar una versión funcional del sistema en una plataforma de acceso online, para evaluación del stakeholder (Fernando Puricelli).

3. Revisión y ajustes con Stakeholder

Presentar una versión con los tres proyectos integrados a el stakeholder, registrar sus comentarios y determinar los agregados necesarios para la implementación del sistema que se desplegará una vez finalizado el desarrollo.

4. Hosting del proyecto

Definir la infraestructura para alojar el proyecto con el fin de dejar la aplicación online.

5. Implementación de mejoras adicionales

Evaluar la introducción de mejoras adicionales coordinadas con el stakeholder durante esta fase de revisión.

6. Creación del manual de despliegue

Elaborar un manual que guíe el proceso de despliegue y configuración del sistema entregado. Coordinar con el equipo de sistemas de la Universidad para revisar los entregables.

Requerimientos funcionales

En el contexto de la constante evolución tecnológica y la necesidad de eficiencia en la gestión universitaria, se ha identificado la necesidad de optimizar y modernizar diversos aspectos del sistema informático actual. La presente serie de requerimientos funcionales se originan en la búsqueda de soluciones que permitan potenciar la experiencia de usuarios clave, así como mejorar la eficacia en la administración de procesos académicos.

Estos requerimientos son presentados por el stakeholder impulsor de este proceso de mejora. Los actores involucrados incluyen a los usuarios finales, directivos y al personal con roles administrativos, todos bajo la premisa de mejorar eficacia de la gestión de equivalencias.

1. Reset de contraseñas de usuarios: Proponer y desarrollar herramientas para que los usuarios puedan autogestionarse las contraseñas de inicio de sesión en caso de que las pierdan u olviden.
2. Activar o desactivar usuarios: El usuario superusuario, debe tener la posibilidad de poder cambiar el estado de los usuarios, con el fin de activar o desactivar su ingreso a la plataforma.
3. Comunicación mejorada del directivo mediante correo electrónico: Durante la revisión de equivalencias ingresadas, el directivo debe tener la posibilidad enviar un comentario en

cada cambio realizado en la equivalencia y que las mismas lleguen al alumno mediante correo electrónico.

4. Mejorar el método de asignación de una carrera a los directivos: Mejorar e implementar una herramienta que permita asignar y cambiar de carreras a los directivos de las mismas, desde el rol de superusuario.
5. Buscador de antecedentes de equivalencias: Mediante la búsqueda de materias aprobadas mediante las universidades de origen se visualizan las equivalencias anteriormente aprobadas.

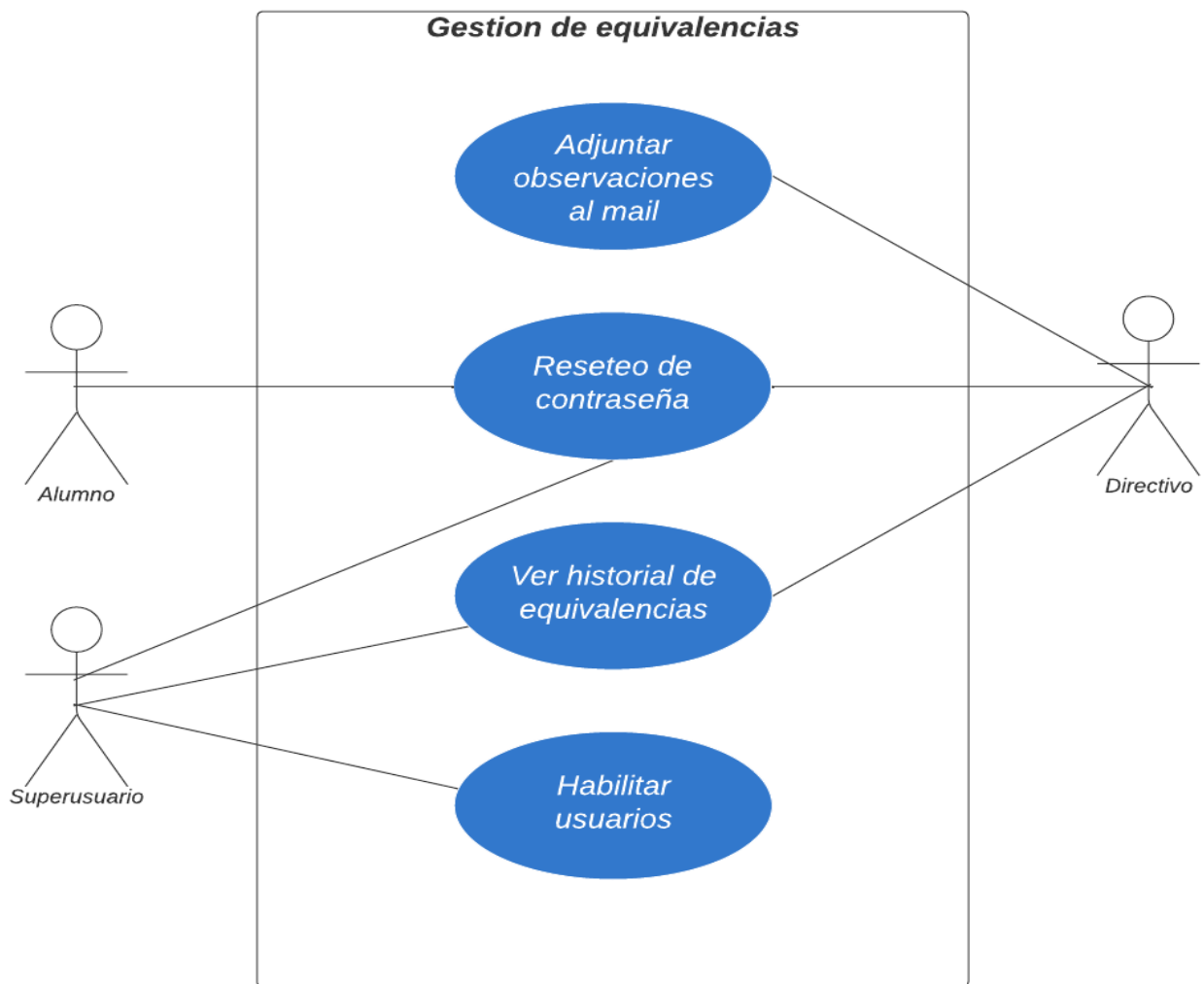
Requerimientos no funcionales

Continuamos con nuestra idea de buscar mejorar y optimizar el proyecto ya integrado, se han detectado requerimientos no funcionales, que no son funciones específicas del sistema, sino propiedades del mismo, tales como:

1. Usabilidad:
 - a. Cambiar el orden en el que se visualiza la información en el perfil de los usuarios al generar o revisar una solicitud de equivalencia: A fin de igualar el aspecto de la aplicación, respecto al documento oficial que se tiene en la Universidad, se solicita el cambio de cómo se visualiza la información en pantalla.
 - b. Mejorar el diseño responsive de la pantalla de login: Se observa que la pantalla no se adapta a los distintos tamaños, se mejora este aspecto especialmente para ofrecer una experiencia óptima en dispositivos móviles.
 - c. Poder visualizar los directivos asignados a una carrera en el apartado “carreras”.
2. Seguridad:
 - a. En el proceso de reset de contraseñas se valida además del hash, enviado por correo electrónico al usuario que pide el reset, el DNI de éste para completar el proceso.
3. Rendimiento:
 - a. Durante el login de los usuarios, inicialmente venía implementada en las versiones recibidas, que para poder autenticar al usuario que está por ingresar al sistema, desde el backend, se enviaba al frontend una lista con todos los usuarios

existentes. Esto se optimizó para poder solo traer la información del usuario que ingresara.

Diagrama de caso de uso



Plan de trabajo

El plan comenzó con un análisis de las tres versiones del proyecto, las tareas de instalación se definieron por separado en cada una de las versiones del proyecto.

Para poder llevar adelante la planificación del proyecto se gestionaron reuniones de planning cada 3 semanas y reviews semanales (de estos casos quedaron registros en las distintas fichas de inicio y fin de sprint respectivamente), en las cuales se pudiese conversar libremente sobre lo pendiente y lo que viene por delante en cada sprint junto con los profesores. También hubo reuniones semanales, de dos a tres, según el nivel de complejidad que tuviese el sprint en curso. Para la organización de tareas se utilizó la metodología *Kanban* (La metodología Kanban se basa en una filosofía centrada en la mejora continua, donde las tareas se extraen de una lista de acciones pendientes en un flujo de trabajo constante mediante tableros) mediante la herramienta llamada *Trello*, Trello que nos permitió visualizar las tareas en formatos de tarjetas y columnas. *Agile* (metodología de gestión de proyectos que se enfoca en la colaboración, adaptabilidad y entrega incremental) ha sido la metodología de trabajo utilizada debido a que todos los integrantes del proyecto nos hemos encargado de resolver las tareas a medida que se presentaban y nos permitía obtener resultados a corto plazo.

Columnas utilizadas para Trello:

Backlog Current Sprint - Para tareas pendientes de asignación para el próximo Sprint

To do - Para tareas pendientes de asignación en el Sprint actual

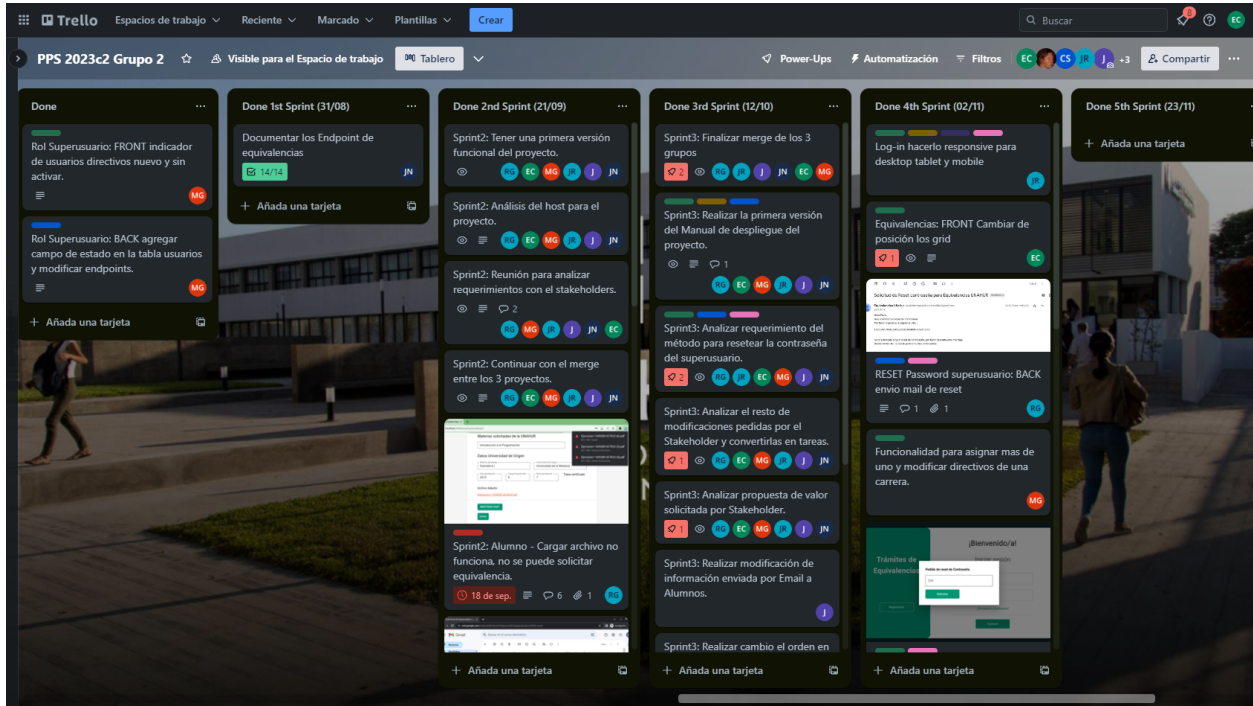
Doing - Para tareas asignadas en proceso

On Hold - Para tareas en stand-by

QA - Para tareas pendientes de testing

Done - Para tareas finalizadas en el Sprint actual

Done Sprint - Para tareas finalizadas en cada Sprint correspondiente



Ficha de Inicio Sprint

Práctica Profesional Supervisada - 2do cuatrimestre 2023

Ficha de principio de sprint

Grupo 2 - Proyecto Equivalencias

Nro de sprint	Fecha de comienzo	Fecha de fin
3	21/09/2023	12/10/2023

Objetivos que nos ponemos para este sprint.

- Finalizar merge de los 3 grupos
- Realizar la primera versión del Manual de despliegue del proyecto.
- Analizar requerimiento del método para resetear la contraseña del superusuario.
- Analizar propuesta de valor solicitada por Stakeholder.
- Analizar el resto de modificaciones pedidas por el Stakeholder y convertirlas en tareas.
- Realizar cambio el orden en lo que se visualizan los datos: primero Materias origen, luego materias unahur.
- Realizar modificación de información enviada por Email a Alumnos.

Práctica Profesional Supervisada - 2do cuatrimestre 2023

Ficha de fin de sprint

Grupo 2 - Proyecto Equivalencias

Nro de sprint	Fecha de comienzo	Fecha de fin
3	21/09/2023	12/10/2023

Objetivos que completamos para este sprint.

- Finalizar merge de los 3 grupos
- Realizar la primera versión del Manual de despliegue del proyecto.
- Analizar requerimiento del método para resetear la contraseña del superusuario.
- Analizar propuesta de valor solicitada por Stakeholder.
- Analizar el resto de modificaciones pedidas por el Stakeholder y convertirlas en tareas.
- Realizar cambio el orden en el que se visualizan los datos: primero Materias origen, luego materias unahur.
- Realizar modificación de información enviada por Email a Alumnos.

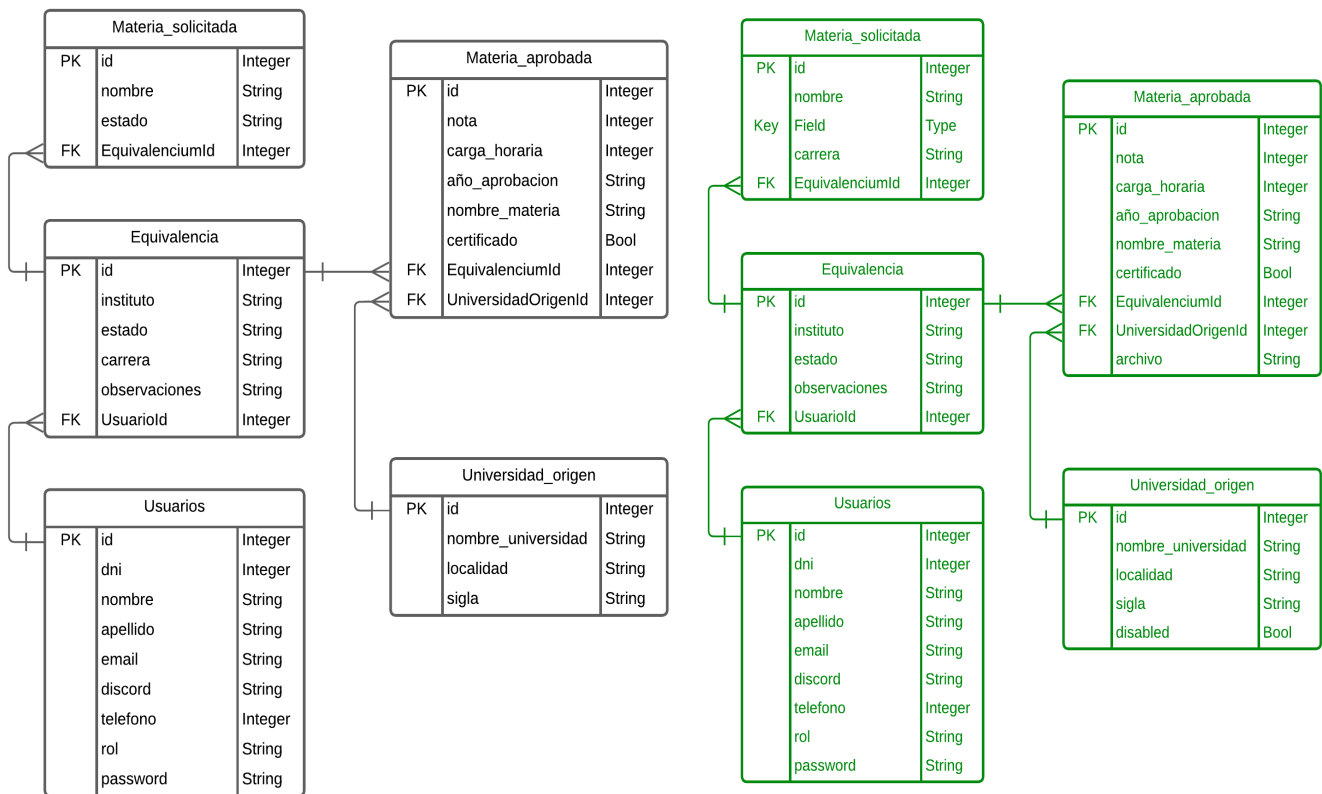
A fin de detectar posibles fallas, se tomaron las versiones de los grupos desde donde partimos con el proyecto, realizamos las instalaciones por separado y se revisaron las funcionalidades de cada una, se buscó entender el modelo de datos y se avanzó en la comprensión de la metodología de desarrollo y las posibles incompatibilidades que se encontrarán durante el proceso de integración. Este primer paso tenía como objetivo principal comprender el funcionamiento de cada versión y las funcionalidades que habían sido implementadas.

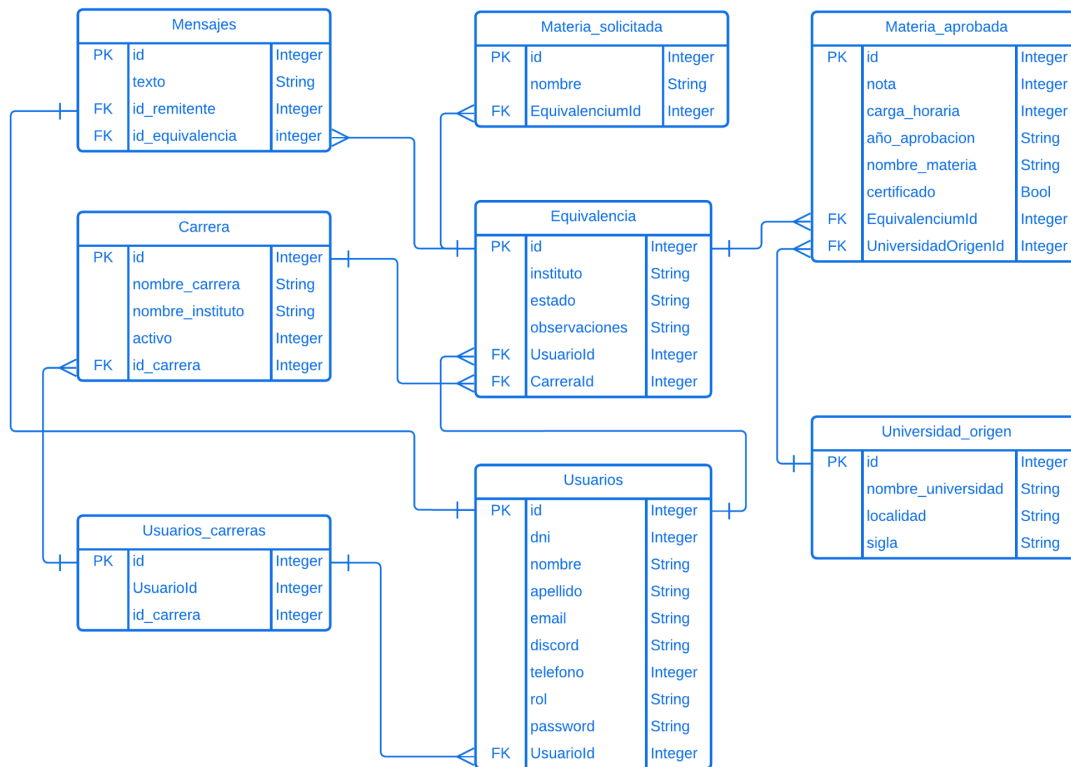
Se identificaron las secciones con similares funcionalidades y las que tenían discrepancias en su totalidad. Las estrategias que surgieron de este primer análisis permitieron definir los pasos a

seguir para integrar el front-end de la aplicación, otros tantos para el back-end y otros para el modelo de datos.

Integración inicial

Relevada la situación inicial, se procedió a integrar las diferentes bases de datos, se utilizó la versión del Grupo 4 (extensión 2) como base, luego se migraron los campos que compatibilizan este modelo con el del Grupo 1 (extensión 1), por último la migración de las tablas nuevas implementadas por el Grupo 3 (extensión 3). Este modelo permite darle servicio a los tres back-end por separado.

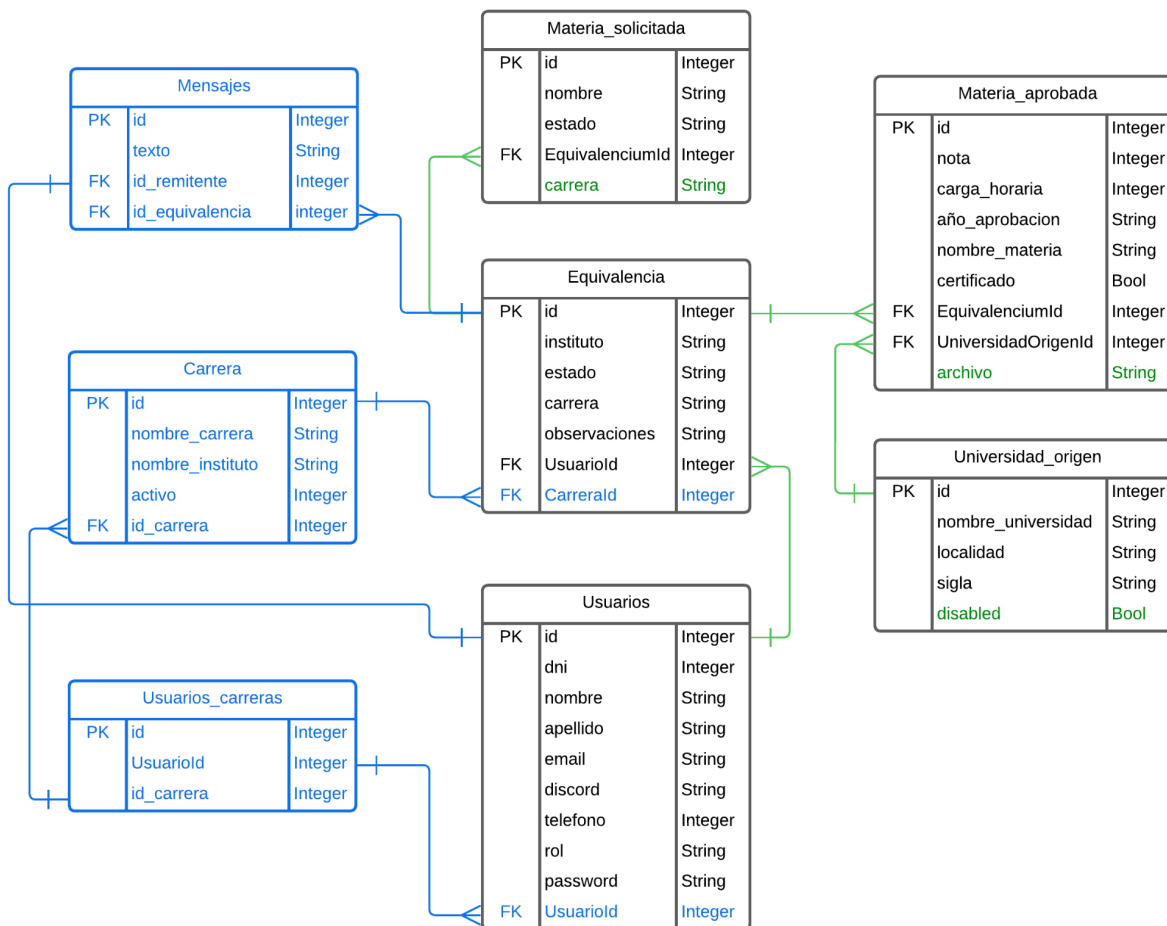




Modelo de datos del Grupo 3

El próximo paso fue integrar el back-end, para ello se utilizó el merge de las ramas de los repositorios del Grupo 4 y Grupo 1, los cuales eran similares. Se detectaron algunas diferencias en funciones de varios endpoints, pero se los compatibilizó según el caso de cada versión. Por último se dejó para agregar manualmente las funciones del Grupo 3, que por cuestiones de compatibilidad, se analizaron función por función hasta lograr que una versión de back-end fuera consistente con las tres versiones.

Integración final



Los 3 modelos de datos integrados

Se dejó para el final la integración de front-end, pero con un orden diferente respecto a lo realizado anteriormente en back-end. La metodología aquí se centró en unir mediante merge, a las versiones del Grupo 3 y Grupo 1, dado que la distribución de componentes y las extensiones estaban similarmente encaradas desde lo funcional. Se dejó para el final la integración manual del Grupo 4, dado que la distribución de su versión se vio más modificada que el resto, debido a que algunos componentes se movieron del sitio original donde habían sido usados en las versiones de los otros Grupos.

Con lo anterior mencionado se alcanzó a una única versión del proyecto con las tres versiones consolidadas, esto sucedió durante la ejecución del sprint 3 del proyecto (documentado más adelante en el cronograma de tareas).

Reunión con el Stakeholder

Una vez que se integraron las extensiones en una única versión del proyecto, se programó una demostración en la UNAHUR para visualizar las funcionalidades en su totalidad con el Stakeholder. Durante la reunión, se exploraron necesidades y requisitos en relación al proceso de gestión de equivalencias académicas. Se recopilaron requisitos adicionales y sugerencias.

Cronograma de tareas

Tareas	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5
Análisis de los proyectos					
Merge de los proyectos					
Primera versión funcional					
Backend					
Frontend					
Reunión con stakeholder					
Manual de despliegue					
Realización de la carpeta					
Demo de presentación					
Presentación final					
Implementación proyecto					

Casos de pruebas

Las pruebas funcionales tienen como finalidad confirmar si el sistema se comporta según lo previsto, se evaluó si cumple o no con sus especificaciones. El plan de pruebas se estructura mediante un conjunto de casos de prueba que se derivan directamente de los requisitos funcionales. Se busca asegurar que cada función o característica del sistema funcione correctamente y de acuerdo con lo que se espera, proporcionando así una validación efectiva del rendimiento del software.

Restablecimiento de contraseña

	Caso de prueba – Restablecer contraseña
Descripción	Generar una nueva contraseña para el usuario.
Condiciones de ejecución	El usuario debe estar registrado en el sistema.
Entrada	Número de DNI
Resultado esperado	Permitir al usuario generar una nueva contraseña para ingresar al sistema.
Evaluación de la prueba	Contraseña restablecida.



Trámites de Equivalencias


[Registrarse](#)

¡Bienvenido/a!

Iniciar sesión

DNI

Contraseña

 [¿Olvidaste tu contraseña?](#)

[Ingresar](#)

Pedido de reset de Contraseña

[Solicitar](#)

Solicitud de Reset contraseña para Equivalencias UNAHUR ▶ Recibidos x



Equivalencias UNaHur <equivalenciasunahuruniversidad@gmail.com>

para mí ▾

Hola Emir,

Has solicitado un reset de contraseña.

Por favor, ingresa a la siguiente URL :

<http://localhost:3000/resetPassword/149755048048048048957250856654149553250>

Si no solicitaste ningún reset de contraseña, por favor descarta este mensaje

Desde UNAHUR, nunca te pediremos tus contraseñas.

← Responder

→ Reenviar

Restablecer Contraseña

Confirmar



Contraseña modificada con éxito





Habilitar o deshabilitar usuarios

	Caso de prueba – Habilitar o deshabilitar usuarios
Descripción	El superusuario puede habilitar y deshabilitar las cuentas de usuarios registrados.
Condiciones de ejecución	El usuario a habilitar o deshabilitar debe estar registrado.
Entrada	Apartado “usuarios”
Resultado esperado	Permitir al superusuario habilitar o deshabilitar usuarios registrados (directivo o alumno)
Evaluación de la prueba	Usuarios habilitados o deshabilitados para su ingreso a la aplicación

Usuarios

Buscar nombre o DNI

Rol

Nombre usuario	DNI	Email	Telefono	Rol	Estado
Liam Gallagher	35025698	Liam595@hotmail.com	1123968547	alumno	 <input checked="" type="checkbox"/> Habilitado
Carlos Rey	35563675	carlos@gmail.com	1190595568	alumno	 <input type="checkbox"/> Habilitado

Búsqueda de equivalencias ya existentes

	Caso de prueba – Buscar equivalencias ya existentes
Descripción	El directivo puede buscar equivalencias ya aprobadas de una universidad y materia específica.
Condiciones de ejecución	Debe haber equivalencias cargadas
Entrada	Universidad de origen y materia aprobada.
Resultado esperado	El directivo debe poder ver las equivalencias relacionadas a los campos ingresados.
Evaluación de la prueba	Aparición en pantallas de las equivalencias relacionadas con la búsqueda.

Solicitudes de equivalencias

Busqueda de equivalencias

Universidades: Universidad Tecnológica Na...
Materias: Gramática I

BUSCAR

Buscar DNI

DNI	Solicitante	Materias Solicitadas	Fecha	Estado	Acciones
30563652	Enzo Fernandez	Introducción a la Programación	9/11/2023	PENDIENTE	

Materias aprobadas en equivalencias de Universidad de Buenos Aires

Listado de materias

	Materia aprobada	Carrera Origen	Última actualización
▼	matematica	informatica	2023-11-12T22:42:36.817Z
▼	intro	informatica	2023-11-12T22:46:43.477Z
^	matematica 2	ingenieria	2023-11-13T19:23:37.153Z

Detalles

Materia solicitada	Estado	Carrera
matematica 2	pendiente	Tec. en Metalurgica











▼	Algoritmos	Informatica	2023-11-14T00:21:34.278Z
---	------------	-------------	--------------------------

Cerrar

Agregar varios directivos a una carrera

	Caso de prueba – Agregar varios directivos a una carrera
Descripción	El superusuario pueda asignar más de un directivo al crear una carrera
Condiciones de ejecución	Debe haber directivos cargados previamente en el sistema
Entrada	Nombre de la carrera, nombre del instituto y directivos a asignar
Resultado esperado	Los directivos deben quedar asignados a una carrera
Evaluación de la prueba	Carrera dada de alta con la asignación de directivos correspondientes

← Carreras → **AGREGAR CARRERA**

Carrera	Instituto	Fecha Actualizacion	Directivo	
Tecnicatura en informatica	Instituto de Informática	9/11/2023 - 8:27 PM	Anthony kiedis	 
Profesorado de Ingles	Instituto de Lenguas	9/11/2023 - 8:27 PM	Anthony kiedis	 
Lic. en Educacion	Instituto de Educación	9/11/2023 - 8:27 PM	Anthony kiedis	 
Lic. en Biotecnologia	Instituto de Biología	9/11/2023 - 8:27 PM	Ana Gonzalez	 
Tec. en Metalurgia	Instituto de Metalurgia	9/11/2023 - 8:27 PM	Ana Gonzalez	 



Agregar Carrera

Nombre de la Carrera

Nombre del Instituto

Directivo/s

AGREGAR **CANCELAR**

Tecnicatura en Informatica	Instituto de Tecnologia	14/11/2023 - 12:17 PM	Ana Gonzalez, Juan Perez	 
----------------------------	-------------------------	-----------------------	--------------------------	---

Entrega y despliegue

Despliegue local:

Para llevar adelante el despliegue local en primer lugar hay que clonar los repositorios correspondientes a las mejoras realizadas en el segundo cuatrimestre del año 2023.

Este procedimiento se puede llevar adelante desde una consola “CMD” o “BASH” depende el sistema operativo.

Pre requisitos:

- Tener instalado GIT
- Tener instalado NodeJS en su versión 14.15.x (hay un programa para controlar las versiones de node que se llama **npm**(node version manager) para manejar lo dicho).
- Tener una conexión a internet.

Para el front-end:

<https://github.com/DesApp-2023c1-Grupo-2/PPS-2023c2-Grupo2-equiv-front.git>

Para el back-end:

<https://github.com/DesApp-2023c1-Grupo-2/PPS-2023c2-Grupo2-equiv-back.git>

Copiar la dirección del repositorio y ubicarse en la consola en el directorio donde se quiere clonar el repositorio.

Ejecutar:

Estos comandos son comunes para los 2 proyectos:

git clone “acá la URL del proyecto” sin las comillas.

entrar a la carpeta del proyecto con (cd PPS-2023c2-Grupo2-equiv-front) o (cd PPS-2023c2-Grupo 2-equiv-back)

npm install (este comando instalará todas las dependencias necesarias para poder ejecutar el proyecto).

Aunque el proceso es automático puede haber dependencias que requieren de una instalación manual, ejecutando los siguientes comandos.

Exclusivos front-end:

```
npm install @mui/material @emotion/react @emotion/styled --save
```

```
npm install mui-image --save
```

```
npm install @material-ui/core@^4.11.2 react@^17.0.0 --save
```

```
npm install bcryptjs --save
```


npm install crypto-browserify --save
npm install @mui/lab@5.0.0-alpha.88 --save

Exclusivos back-end:

npm install fs path --save
npm install multer --save
npm install bcryptjs --save
npm install crypto-browserify --save

Crear un archivo en el directorio raíz llamado .env, dentro de este archivo se van a declarar las siguientes variables de entorno.

```
SQL_USERNAME=postgres  
SQL_PASSWORD=admin  
SQL_DATABASE=equivalencias
```

```
NODEMAILER_USER='email@gmail.com' ← mail de la institución  
NODEMAILER_PASSWORD='nzdpd frgy frih gylc' contraseña del mail
```

```
SQL_TEST_DATABASE=equivalencias
```

El paso siguiente es la creación de la base de datos con la que va a trabajar el proyecto, en este caso se utiliza el motor de base de datos **PostgreSQL**. En este punto contamos con 2 opciones para crearla.

Mediante instalación local:

Esta es la forma clásica, se instala **PostgreSQL** en la **PC**, luego de instalar y tener el programa gestor de bases de datos ejecutado (PGAdmin por defecto de postgres), se seleccionó en crear nueva base de datos se despliega un asistente en donde se ingresan los datos para la creación, nombre de la DB, nombre de usuario, contraseña y puerto (el default es 5432), estos datos son los que se van a pasar al back-end para conectarse. También se puede hacer por consola con los siguientes comandos.

Postgres:

```
CREATE USER postgres WITH PASSWORD admin;  
CREATE DATABASE equivalencias; ← “nombre_base_de_datos”
```

Conectarse a la base de datos:

```
C:\Users\ce>psql -U postgres -W -p 5432 localhost equivalencias (para windows)  
sudo -i -u postgres (para linux)
```

—

- U nombre del usuario con el que nos vamos a conectar
- p puerto de conexión (por defecto es el 5432)
- 'equivalencias' es el nombre de nuestra base de datos.
- localhost (donde se expone la DB)

Otra alternativa a la base de datos es con **Docker** que nos permite tener una base de datos totalmente funcional sin tener que instalar **PostgreSQL** en la PC, siguiendo los siguientes pasos. Desde el microsoft store en búsqueda si ingresan linux van a tener muchas distribuciones de linux pueden elegir la que mas les guste, o mediante la consola con el siguiente comando.

wsl --set-version Ubuntu-20.04 2

y se instala un entorno virtual de ubuntu 20.

Dentro del mismo hay que instalar **Docker** y ejecutar los siguientes comandos.

Add Docker's official GPG key:

sudo apt-get update

sudo apt-get install ca-certificates curl gnupg

sudo install -m 0755 -d /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

sudo chmod a+r /etc/apt/keyrings/docker.gpg

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

Una vez instalado docker se crea una base de datos **PostgreSQL** con los datos de conexión necesarios con el siguiente comando.

docker run --name pps -e POSTGRES_USER=postgres -e POSTGRES_PASSWORD=admin -e POSTGRES_DB=equivalencias -p 5432:5432 -d postgres:12.5-alpine

--name pps ← nombre del contenedor

-e POSTGRES_USER=postgres

-e POSTGRES_PASSWORD=admin

-e POSTGRES_DB=equivalencias

-p 5432:5432 ← puerto expuesto por el contenedor

postgres:12.5-alpine ← imagen postgres para crear el contenedor.

Este comando va a descargar la imagen de postgres y creará un contenedor con los datos que le pasamos. al terminar con solo ejecutamos el contenedor con:

sudo docker start pps ← **pps** es el nombre que le pasaron al crear el contenedor.

sudo docker stop pps ← detiene el contenedor(la base de datos ya no está disponible, pero los datos persisten).

Para gestionar la base de datos se puede utilizar cualquier programa para tal fin como por ejemplo **DBeaver**, **PGAdmin**, etc.

Una vez esté la base de datos disponible, en el directorio raíz del proyecto de back-end, se ejecuta el comando **npm run db:init** y luego **npm run db:seed**. Estos comandos crean las tablas de la base de datos y crea un superusuario que es el usuario que va a administrar la aplicación.

Una vez terminado el proceso se inicia la ejecución con **npm start**. Este comando levanta el back-end y lo deja a la espera de peticiones.

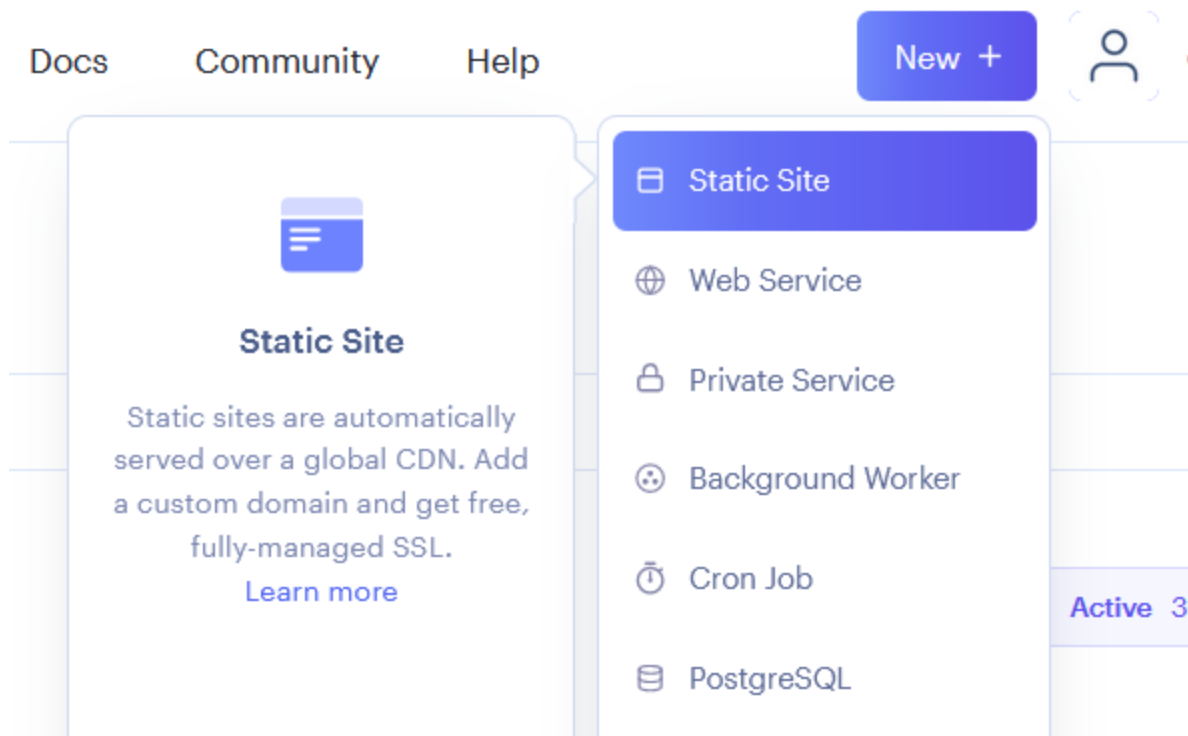
Para el front-end al ejecutar el comando **npm start**, lanza la aplicación en el browser por defecto en el **localhost**, y de esta forma termina el proceso de despliegue de la aplicación en el entorno local.

Despliegue remoto:

El desarrollo del despliegue de la aplicación se realiza a partir de 2 repositorios alojados en github. Uno para el frontend y otro para el backend.


El despliegue se puede realizar completamente en el HOST RENDER.

Despliegue del frontend : Dentro de render se elige nuevo proyecto static site.



Luego hay que elegir un proyecto de los que se tengan en github.

Connect a repository

 DesApp-2023c1-Grupo-2 / PPS-2023c2-Gurpo2-equiv-front • 3 hours ago Connect

 DesApp-2023c1-Grupo-2 / PPS-2023c2-Gurpo2-equiv-back • a day ago Connect

Al conectarse con el proyecto se cargan los datos correspondientes al proyecto.

You are deploying a static site for **DesApp-2023c1-Grupo-2/PPS-2023c2-Gurpo2-equiv-front**.

You seem to be using `create-react-app`, so we've autofilled some fields accordingly. Make sure the values look right to you!

Name A unique name for your static site.	<input type="text" value="example-service-name"/> Required
Branch The repository branch used for your static site.	<input type="text" value="master"/>
Root Directory <small>Optional</small> Defaults to repository root. When you specify a root directory that is different from your repository root, Render runs all your commands in the specified directory and ignores changes outside the directory.	<input type="text" value="e.g. src"/>
Build Command This command runs in the root directory of your repository when a new version of your code is pushed, or when you deploy manually. It is typically a script that installs libraries, runs migrations, or compiles resources needed by your app.	<input type="text" value="\$ yarn build"/>
Publish directory The relative path of the directory containing built assets to publish. Examples: <code>./</code> , <code>./build</code> , <code>dist</code> and <code>frontend/build</code> .	<input type="text" value="build"/>

Advanced ▾

Create Static Site

- Nombre del proyecto y el que va a tener en la dirección web.
- Branch: Rama a la que queremos hacer el despliegue.
- Root directory: Si queremos especificar un root específico para la aplicación.
- Build command: Comando con el cual se construye la aplicación.
- Publish directory: Carpeta en donde se van a alojar los datos de la aplicación.

Para desplegar esta aplicación es necesario utilizar los siguientes datos:

- Name: **unahur-equivalencias**
- Branch: **deploy-branch.**
- Root directory:
- Build command: **npm run build**
- Publish directory: **build**

Advanced ^

Use environment variables to store API keys and other configuration values and secrets. You can access them in your code like regular environment variables, for example with `os.getenv()` in Python or `process.env` in Node.

Add Environment Variable

You can store secret files (like `.env` or `.npmrc` files and private keys) in Render. These files can be accessed during builds and in your code just like regular files.

Secret files are available to read at the root of your repo (or Docker context). They are also available to load by absolute path at `/etc/secrets/<filename>`.

Add Secret File

Auto-Deploy

Automatic deploy on every push to your repository or changes to your service? Select "No" to handle your deploys manually.

Yes

Build Filters

Filter for files and paths to monitor for automatic deploys. Learn about [build filters](#). **Filter paths are absolute**. If you've defined a root directory, you can still define paths outside of the root directory.

Included Paths

Changes that match these paths will trigger a new build.

+ Add Included Path

Ignored Paths

Changes that match these paths will **not** trigger a new build.

+ Add Ignored Path

En opciones avanzadas tenemos la posibilidad de agregar variables de entorno y también indicar las versiones de las dependencias que se van a instalar. Esto va a obviar las que vienen en el package.json del repositorio.

Key	Value	
DATABASE_URL	🗑️
PORT	🗑️
SQL_DATABASE	🗑️
SQL_PASSWORD	🗑️
SQL_USERNAME	🗑️

Environment Variables:

CI: False
REACT_APP__URL: “aca va la URL que expone el back-end”

Le aplicamos el siguiente redireccionamiento:

Source: /* Destination: /index.html Action: Rewrite

Buttons: Add Rule, Save Changes

Paso siguiente crear el despliegue:
El servidor comenzará a desplegar la aplicación de acuerdo con los parámetros suministrados. En una ventana bash se pueden ver los logs que son los mismos que los mostrados al iniciar la aplicación localmente.

November 13, 2023 at 7:19 PM Building Cancel deploy

badafe5 Merge pull request #1 from DesApp-2023c1-Grupo-4/Dev-Fr...

Search logs Search Maximize Scroll to top

```
Nov 13 07:19:47 PM ==> Cloning from https://github.com/DesApp-2023c1-Grupo-2/PPS-2023c2-Garpo2-equiv-front...
Nov 13 07:19:48 PM ==> Checking out commit badafe50db9eee81051715cdf35d3531b70128fd in branch DesApp-2023c1-Grupo-4
Nov 13 07:19:51 PM ==> Requesting node version lts/fermium
Nov 13 07:19:52 PM ==> Using Node version 14.21.3 via /opt/render/project/src/.nvmrc
Nov 13 07:19:52 PM ==> Docs on specifying a Node version: https://render.com/docs/node-version
Nov 13 07:19:55 PM ==> Using Ruby version 3.2.2 (default)
Nov 13 07:19:55 PM ==> Docs on specifying a Ruby version: https://render.com/docs/ruby-version
Nov 13 07:19:56 PM ==> Installing dependencies with npm...
```

Un mensaje notifica que la aplicación está online.

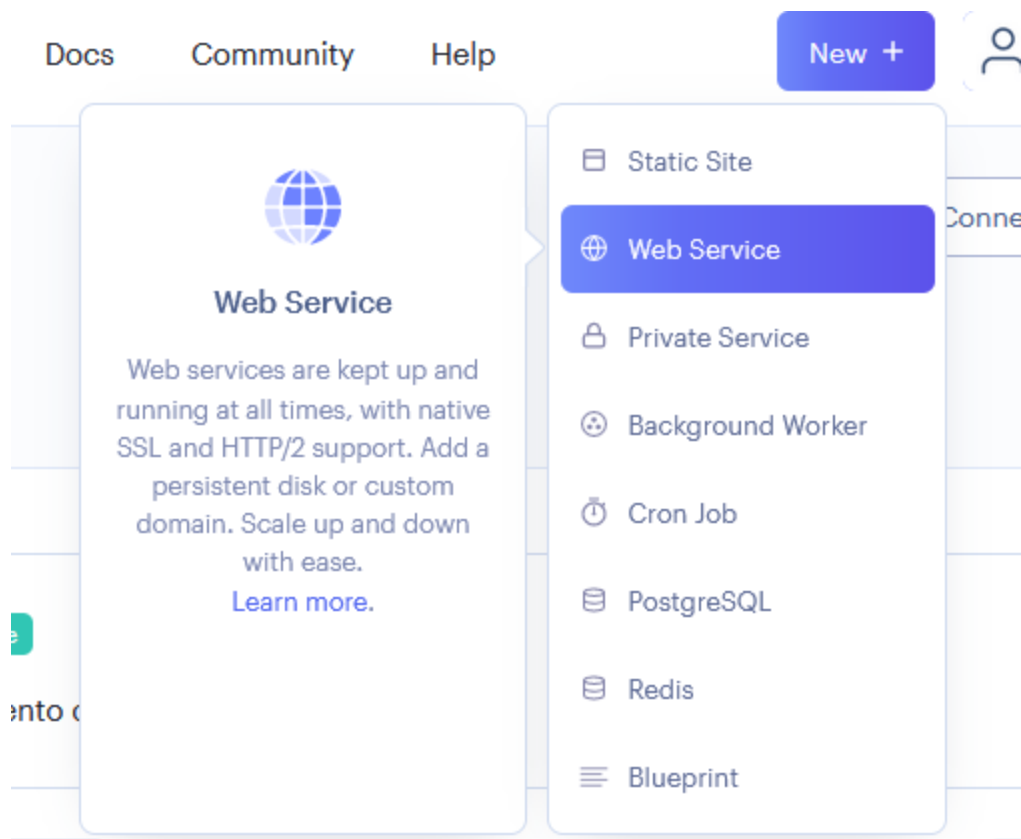
November 13, 2023 at 4:14 PM Live

2b92793 archivo de redireccionamiento de url para react.

Search logs Search Maximize Scroll to top

```
Nov 13 04:17:42 PM
Nov 13 04:17:42 PM https://cra.link/deployment
Nov 13 04:17:42 PM
Nov 13 04:17:45 PM ==> Uploading build...
Nov 13 04:17:50 PM WARNING: Python 3.7.x is no longer officially supported by the Google Cloud CLI
Nov 13 04:17:50 PM and may not function correctly. Please use Python version 3.8 and up.
Nov 13 04:17:50 PM
Nov 13 04:17:50 PM If you have a compatible Python interpreter installed, you can use it by setting
Nov 13 04:17:50 PM the CLOUDSDK_PYTHON environment variable to point to it.
Nov 13 04:17:50 PM
Nov 13 04:17:50 PM ==> Your site is live 🎉
```

Para el back es lo mismo pero se elige la opción de **servicio web**:



Al igual que en el ejemplo anterior tenemos que ingresar los datos correspondientes a las variables de entorno.

DATABASE_URL: "Aquí va la URL que expone la base de datos"

PORT: "Definimos el puerto si queremos, de lo contrario se asigna automáticamente"

SQL_DATABASE: "El nombre de la base de datos creada"

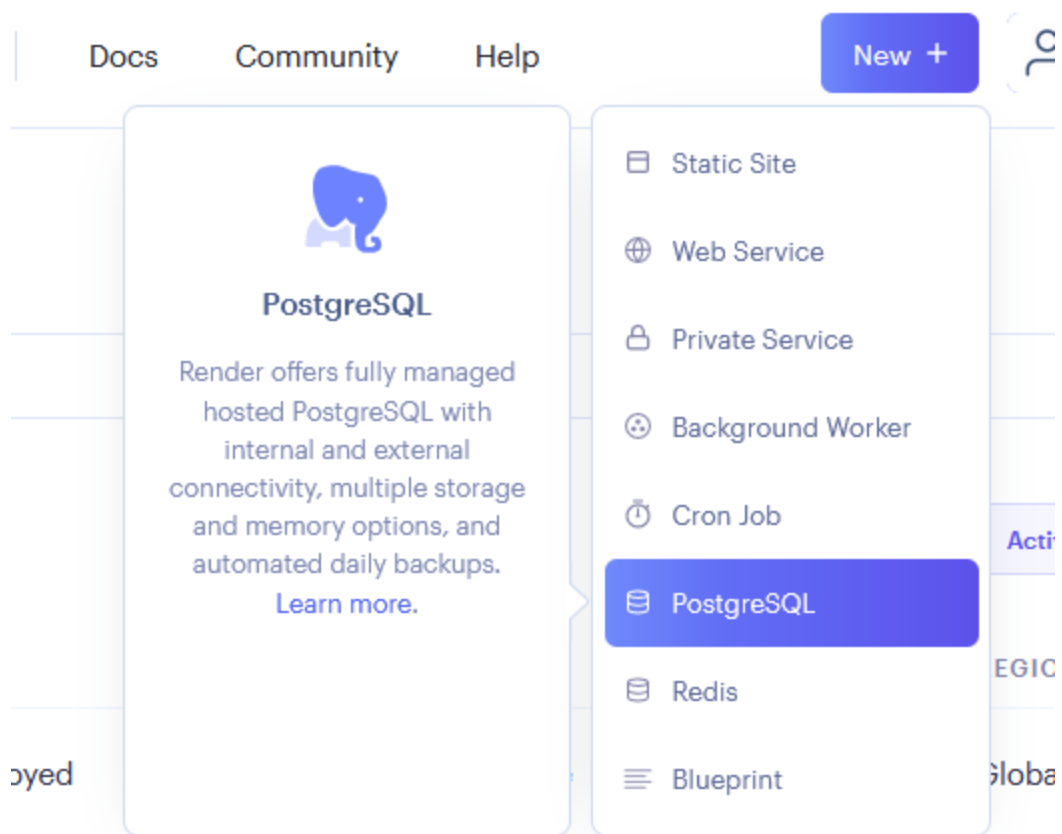
SQL_PASSWORD: "Contraseña de la base de datos"

SQL_USERNAME: "Usuario de la base de datos"

Con esto obtenemos el despliegue en el servidor.

Despliegue de la base de datos:

La base de datos, como en el entorno local, hay que tenerla disponible antes de ejecutar el back sino lanzará error. Elegimos la opción para crear uno nuevo, y seleccionamos la base de datos PostgreSQL.



A continuación se describen las características que va a tener la base de datos.

New PostgreSQL

[Read the docs](#)

Name

A unique name for your PostgreSQL instance.

example-postgresql-name

Database Optional

The PostgreSQL "dbname"

randomly generated unless specified

User Optional

randomly generated unless specified

Region

The [region](#) where your PostgreSQL instance runs. Services must be in the same region to communicate privately and you currently have services running in Oregon.

Oregon (US West)

PostgreSQL Version

15

Datadog API Key Optional

The API key to use for sending metrics to Datadog. Setting this will enable Datadog monitoring.

Please [enter your payment information](#) to select an instance type with higher limits.

En la versión gratuita, el nombre y la password no se pueden elegir, solo el usuario. Una vez creada la base de datos, nos brinda toda la información necesaria para poder conectarnos.

General

Name base-pps-equivalencias [Edit](#)

Created 4 days ago

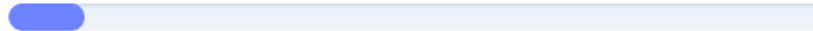
Status ● Available

PostgreSQL Version 15

Region Oregon (US West)

Read Replica [Add Read Replica](#) ⓘ

Storage 6.52% used out of 1.0 GiB



Datadog API Key [Add Datadog API Key](#)

Instance Type Free • RAM 256 MB CPU 100m Storage 1 GB [Update](#)

i A credit card is required to change instance types. [Add payment information](#)

Connections

Hostname ⓘ dpg-cl6iqk2073pc73d9q180-a

Port 5432

Database equivalencias_zrne

Username citorace

Password 📄 👁

Internal Database URL 📄 👁

External Database URL 📄 👁

PSQL Command 📄 👁

Estos son los datos que se necesitan para conectar el back-end. Tenemos dos URL, la interna que sirve para conectarse directamente entre despliegues dentro de el mismo render y la externa que es la que se puede acceder desde cualquier lugar. Se puede conectar a la base de datos con una conexión SSH mediante el putty y un gestor de base de datos para interactuar con la base remotamente.

Finalmente tendremos los proyectos desplegados:

NAME	STATUS	TYPE	RUNTIME	REGION
📁 unahur-equivalencias	✔ Deployed	Static Site	Static	Global
🌐 equiv-back	✔ Deployed	Web Service	Node	Oregon
📁 base-pps-equivalencias	✔ Available	PostgreSQL	PostgreSQL 15	Oregon

Algunos enlaces útiles:

<https://www.youtube.com/watch?v=H1dhlz7zAuo>

<https://www.youtube.com/watch?v=Pjcr7gNOkuU>

https://www.youtube.com/watch?v=KEJ8_yvy0Q&t=818s

<https://render.com/docs/redirects-rewrites>

<https://dev.to/rajeshroyal/page-not-found-error-on-netlify-reactjs-react-router-solved-43oa>

<https://stackoverflow.com/questions/27732209/turning-off-eslint-rule-for-a-specific-line>

https://answers.netlify.com/t/support-guide-i-ve-deployed-my-site-but-i-still-see-page-not-found/125/135?utm_source=404page&utm_campaign=community_tracking

<https://render.com/docs/monorepo-support>

Requerimientos pendientes

1. Agregar la opción de un URL, en contraparte del subir PDF: Al generar una nueva solicitud de equivalencia, el usuario debe contar con la opción de ingresar una URL en vez de un archivo PDF para la carga de archivos.
2. Agregar un campo (combo editable), materias solicitadas de UNAHUR: La materia solicitada debe ser seleccionada de un listado en vez de ingresarla manualmente para evitar errores de carga.