Pedidos de materiales para laboratorio de Biotecnología

Practica Profesional Supervisada

Profesores:
Lombardi, Carlos
Schiffino, Cristian

Alumnos:
Claudio Molina
Leandro de Rivas
Gabriel de Rivas
Nicolas Ramirez
Agustin Fiordalisi

11/07/2024

UNIVERSIDAD NACIONAL DE HURLINGHAM

Instituto de Biotecnología

Índice

Objetivo del proyecto

El objetivo de este proyecto surge de la necesidad del personal del Laboratorio del Instituto de Biotecnología de la Universidad Nacional de Hurlingham (UNAHUR) que tiene como principal tarea la preparación de los materiales como también la asignación del aula para poder realizar los trabajos prácticos y ensayos de los docentes del instituto de biotecnología.

Antecedentes

Necesidades que originan el proyecto

En la actualidad para que un docente pueda reservar un laboratorio y materiales para los trabajos prácticos, debe enviar un email al personal de laboratorio con el detalle de cada uno de los elementos necesarios para las prácticas. Este email se redacta las siguientes características:

- nombre y apellido del docente
- materia
- cantidad de alumnos
- grupos
- fecha y hora deseada para la práctica.
- lista de elementos:
    - nombre de elemento
    - cantidad
    - observación

Luego el laboratorio con el email recibido del docente, asigna el espacio, los equipos y preparaba los materiales y reactivos para dictar la clase. En caso de no contar con algún requerimiento o si el pedido se encuentra incompleto, el laboratorio tiene que contactar por el mismo medio con el que el docente inició el primer contacto para dicho pedido.

Proyecto heredado y nuestras mejoras

A raíz de estas necesidades, en el año 2022 un grupo de Desarrollo de aplicaciones comenzó a trabajar en el desarrollo del sistema que facilitaría la carga y gestión de los pedidos. Este grupo llevó a cabo la realización de una primera versión a lo largo de dos cuatrimestres. Eventualmente nosotros heredamos el proyecto y trabajamos en él durante el transcurso del segundo cuatrimestre del año 2023 y el primero del 2024.

Usuarios

Cambios: Al incorporarnos al desarrollo, el proyecto que recibimos contaba con dos perfiles de usuario, uno para el personal del laboratorio y otro para los profesores. Nosotros identificamos la necesidad de incorporar un nuevo rol que se encargue de gestionar los

Volver

usuarios, ya que en la propuesta inicial cualquier integrante del laboratorio tenía acceso a todas las credenciales existentes en el sistema y podía modificarlas. Para suplir esta problemática, integramos el rol “Admin”, que tiene la facultad de crear nuevos usuarios y modificar los existentes y también es el único que puede visualizar a los demás usuarios.

Beneficios

Detección de vulnerabilidades: Al identificar la necesidad de un nuevo rol, se reconocieron las vulnerabilidades en la gestión de usuarios y la exposición innecesaria de credenciales sensibles. Esta visión permitió diseñar una solución más segura y eficiente.

Mejora en la gestión de usuarios: El rol de "Admin" centraliza la gestión de usuarios, que permite que solo personas autorizadas puedan crear y modificar cuentas. Esto asegura una administración más controlada y organizada.

Reducción de riesgos: Al limitar el acceso a las credenciales y la capacidad de modificarlas solo al rol de administrador, se reduce el riesgo de accesos no autorizados y cambios indebidos. Esto protege la integridad de los datos y minimiza la posibilidad de errores humanos o acciones malintencionadas.

Control de acceso: El rol de "Admin" es el único que puede visualizar y modificar a los demás usuarios, lo que asegura que solo personal capacitado y autorizado tenga acceso a estas funcionalidades críticas.

Seguridad y privacidad: Limitar la visualización de usuarios a un solo rol previene la exposición innecesaria de información sensible, que protege la privacidad de los usuarios y mantiene la confidencialidad de los datos.

Responsabilidad y trazabilidad: Al asignar la gestión de usuarios a un rol específico, es más fácil rastrear quién hizo cambios y cuándo, esto mejora la responsabilidad y la trazabilidad dentro del sistema.

Seguridad

Cambios: Dentro del proyecto se podía iniciar sesión, pero esto era incluso contraproducente, ya que al introducir las credenciales se guardaban en el local storage y tanto el usuario como la contraseña eran visibles en la ruta actual. De igual forma, sin autenticarse se podía acceder a cualquier ruta desde el navegador y visualizar contenidos de otros usuarios registrados e incluso operar sobre ellos como intruso. Sobre este problema incorporamos el encriptado de la información sensible, implementamos el sistema con rutas protegidas (no se puede acceder a rutas para las cuales no se cuente con los permisos necesarios), eliminamos la persistencia de usuarios en el local storage y agregamos la implementación de JWT (Json Web Token) para manejar la sesión.

Volver

Beneficios

Seguridad mejorada: Al eliminar la persistencia de credenciales en el local storage, se reduce significativamente el riesgo de que terceros accedan a información sensible. Las credenciales ya no son visibles en el navegador, que protege así la privacidad del usuario.

Protección de datos: La encriptación asegura que la información sensible esté protegida y solo pueda ser leída por aquellos que tengan la clave adecuada para desencriptarla. Esto ayuda a prevenir el robo de datos y asegura la confidencialidad de la información.

Control de acceso: Con las rutas protegidas, sólo los usuarios autenticados y autorizados pueden acceder a ciertas partes de la aplicación. Esto previene el acceso no autorizado y protege la información y acciones sensibles de ser manipuladas por intrusos.

Mejor experiencia de usuario: Los usuarios tienen la tranquilidad de saber que su información está segura y que no será accedida o modificada por otros sin permiso.

Autenticación segura: JWT permite una autenticación segura y eficiente. Los tokens se emiten después de un inicio de sesión exitoso y contienen la información necesaria para identificar al usuario y sus permisos.

Manejo de sesiones sin almacenamiento inseguro: JWT no requiere almacenar credenciales en el cliente. Los tokens se envían en cada solicitud al servidor, y asegura que solo los usuarios autenticados puedan acceder a las rutas protegidas.

Escalabilidad y flexibilidad: JWT es fácil de implementar y se adapta bien a aplicaciones modernas.

Gestión

Cambios: Tanto la gestión de usuarios, materiales, reactivos y equipos se podían dar de alta, modificar y eliminar permanentemente. Estas funcionalidades contaban con varios problemas, dos de los más destacables eran que varias de las modificaciones que se hacían no se persistían en la base de datos (por lo que al recargar la página estos cambios se perdían) y que esta forma de realizar el borrado de los datos puede generar varias inconsistencias (un ejemplo sería que al eliminar un material, si se consultaba un pedido que lo contenía, se podía observar un espacio vacío donde debía estar el material). Respecto a estos inconvenientes tomamos dos medidas para corregirlos, la primera fue refinar el comportamiento de las modificaciones para que las mismas se persistan en la base de datos, así como también contar con un control de stock de los ítems y con la posibilidad de controlar la cantidad de ítems en estado de reparación. La segunda fue agregar el borrado lógico como herramienta para deshabilitar la disponibilidad de ciertos recursos (con esto podemos evitar, por ejemplo, que ciertos materiales se pidan en la carga de pedidos) sin perder la consistencia de los pedidos ya cargados.

Volver

Beneficios

Consistencia de datos: Refinar el comportamiento de las modificaciones asegura que cualquier cambio realizado por los usuarios se guarde de manera permanente en la base de datos. Esto evita la pérdida de datos y garantiza que las actualizaciones sean efectivas y visibles en todo momento.

Gestión eficiente de recursos: Introducir un control de stock permite monitorear y gestionar los niveles de materiales, reactivos y equipos disponibles. Además, controlar la cantidad de ítems en estado de reparación asegura que estos recursos estén disponibles cuando se necesiten, esto optimiza así la operación del laboratorio.

Reducción de desperdicios: Al tener un control más preciso sobre los recursos, se minimiza el desperdicio y se maximiza la eficiencia en el uso de materiales y equipos.

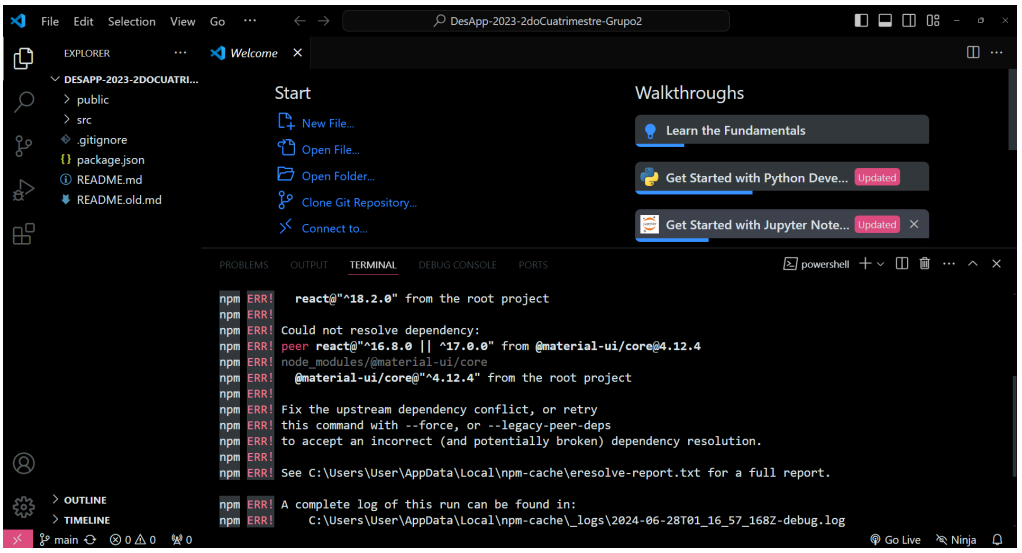
Consistencia en los registros: El borrado lógico permite deshabilitar recursos (como materiales o equipos) sin eliminarlos físicamente de la base de datos. Esto evita inconsistencias al mantener la integridad de los registros existentes, como los pedidos que ya han sido cargados y hacen referencia a estos recursos.

Mejora en la calidad de datos: Al evitar la eliminación física y optar por el borrado lógico, se mantiene la coherencia de la información almacenada, esto proporciona una base de datos más limpia y consistente.

Situación inicial del proyecto

Clonación e instalación del proyecto

Frontend

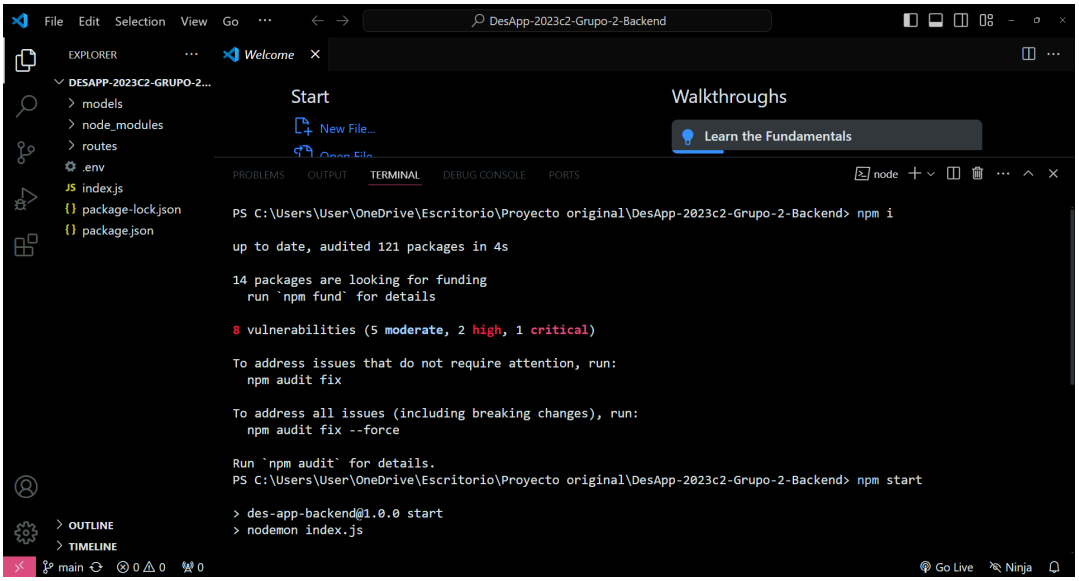


Volver

Al momento de clonar el repositorio **DesApp-2023-2doCuatrimestre-Grupo2** (Frontend) en nuestro entorno local, nos encontramos con que al intentar instalar las dependencias con el comando **npm install** el comando no se ejecutaba correctamente debido a problemas con las versiones de las dependencias indicadas.

Luego de investigar con el equipo notamos que para instalar las dependencias del proyecto del frontend debemos utilizar el comando **npm i --legacy-peer-deps.**

Backend



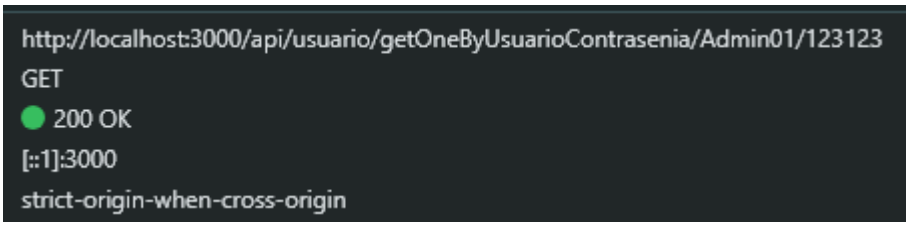
El lado del backend lo levantamos sin problemas con el comando **npm install** y luego **npm start.**

Login

Visualización del Login (proyecto heredado)

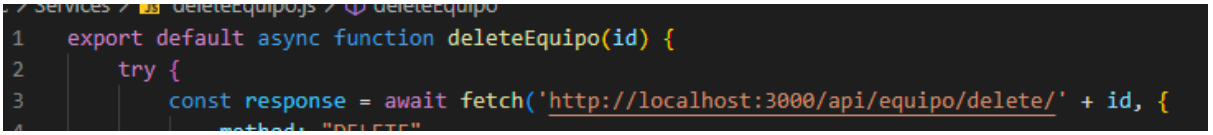
En cuanto al Login heredado observamos que al momento de querer ingresar las credenciales, campos sensibles como la contraseña se envían mediante la url del navegador. Esto lo identificamos como una vulnerabilidad a mejorar del sistema.

Volver



Contraseña enviada por URL

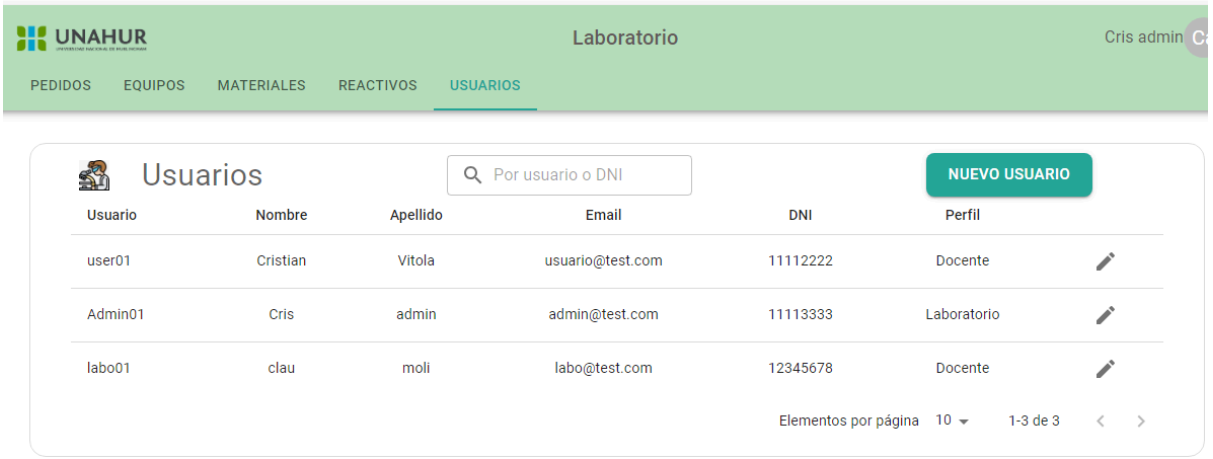
Todos los servicios del frontend tenían la URL base hardcodeada, en lugar de utilizar un archivo de configuración de entorno, lo cual limita la escalabilidad y dificulta realizar cambios.



Visualización de URL Base en los servicios del Frontend

Modificar la URL en el navegador permitía acceder al perfil de otro usuario sin ninguna validación adicional.

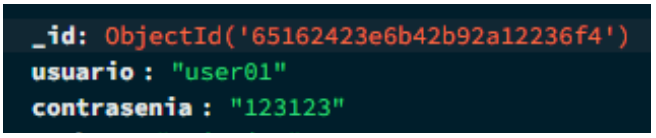
En la pestaña de usuarios, no se realizaba validación alguna, lo que permitía la eliminación accidental del usuario Administrador. Esto resultaba en la imposibilidad de acceder al proyecto de la manera habitual. Además, cualquier usuario independientemente de su rol, podría eliminar a cualquier otro.



Visualización del perfil Admin en la lista de Usuarios (proyecto heredado)

Volver

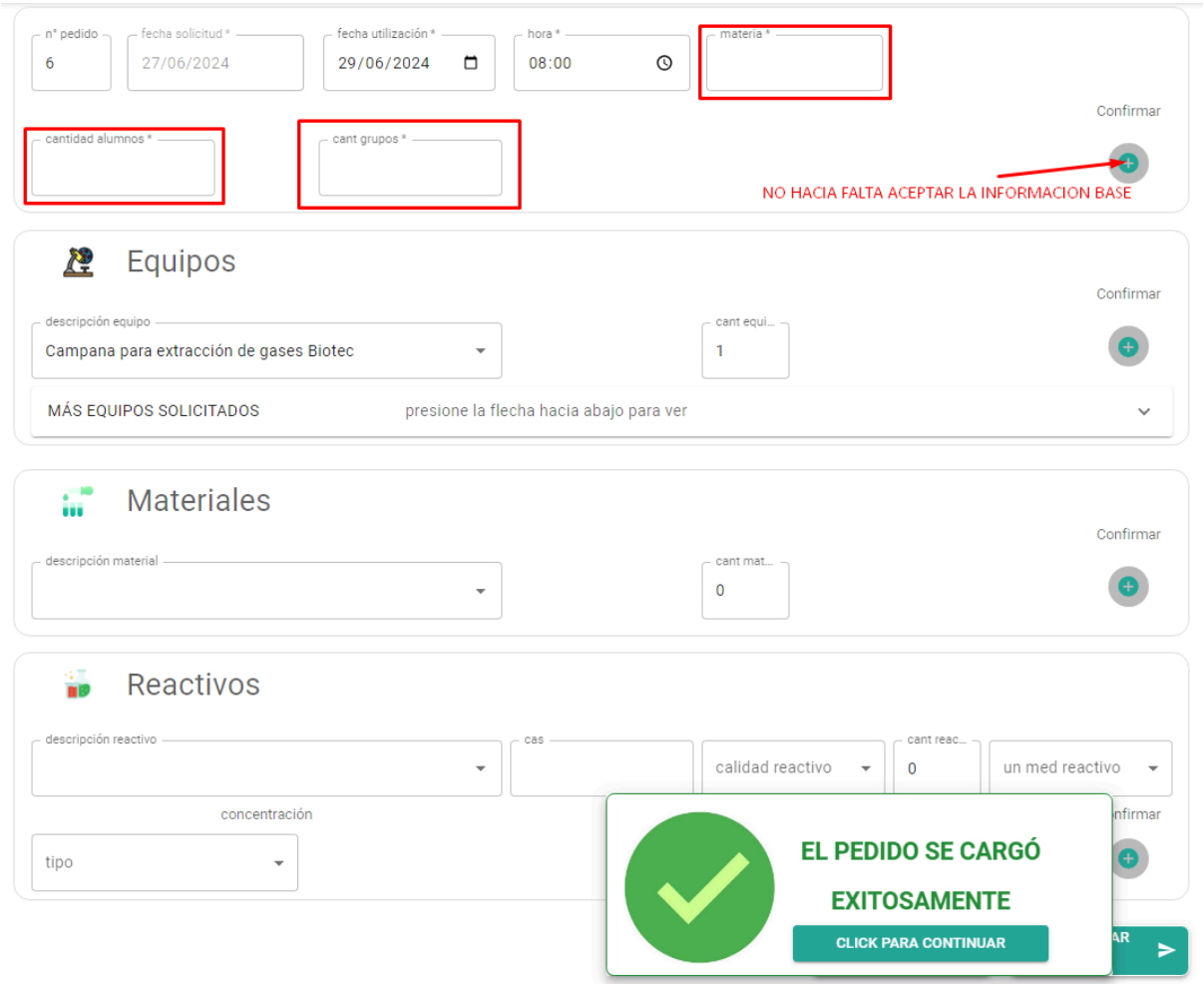
En la siguiente imagen se puede observar que la contraseña guardada en la Base de datos no relacional, se encuentra visible, esto es un error de seguridad, ya que el administrador de la base de datos, podría logear en la cuenta de los usuarios sin problema o simplemente compartir esta información.



contraseña 123123 en Base de datos sin encriptar

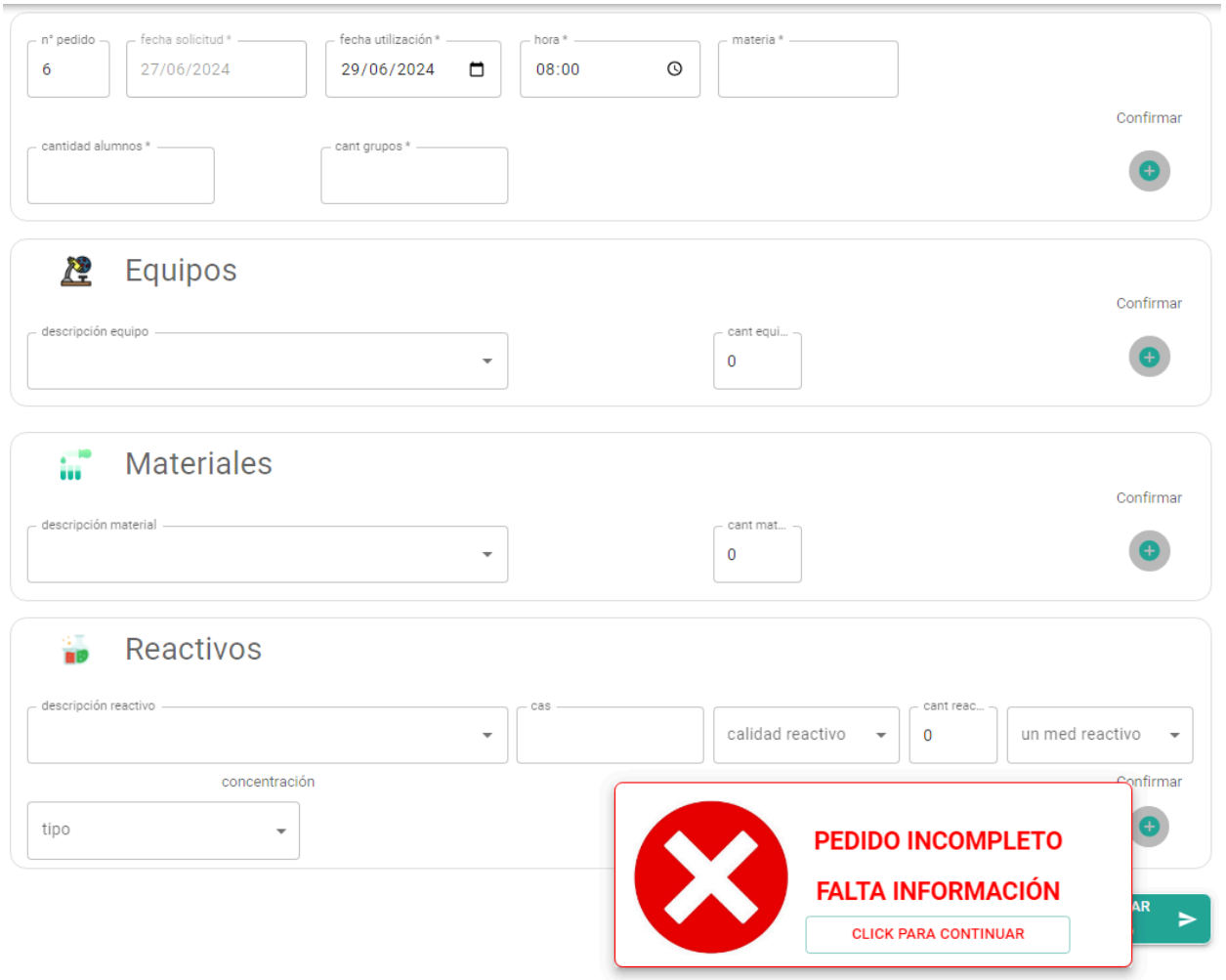
Creación de pedido

En el Front-end de la aplicación, al momento de crear pedidos identificamos la falta de validación en campos esenciales.



Volver

Los mensajes de error no eran descriptivos y no indicaban qué campos debían completarse.



Cada reactivo podía agregarse a un pedido, sin la necesidad de estar completos correctamente.

Volver

Descripción del proyecto

Bajo esta problemática en la cual no había coordinación entre el laboratorio y docente, el equipo anterior construyó una aplicación para gestionar la carga de pedidos de forma centralizada desde un único sistema.

Desde el área de biotecnología se les solicitó expresamente que la aplicación pueda manejar pedidos que incluyan tres tipos de categorías.

Estas categorías son:

Equipos: maquinarias utilizadas para manipular reactivos en un ambiente seguro y bajo condiciones controladas. Dado que suelen ser de gran tamaño, su traslado no es posible, lo que influye en la selección del laboratorio para llevar a cabo las prácticas.

Ejemplo de equipo instalado en un laboratorio.

Materiales: principalmente objetos de vidrio que sirven para contener, medir, calentar, separar, inocular y mezclar reactivos, entre otras funciones.

Ejemplo de material fácilmente trasladable.

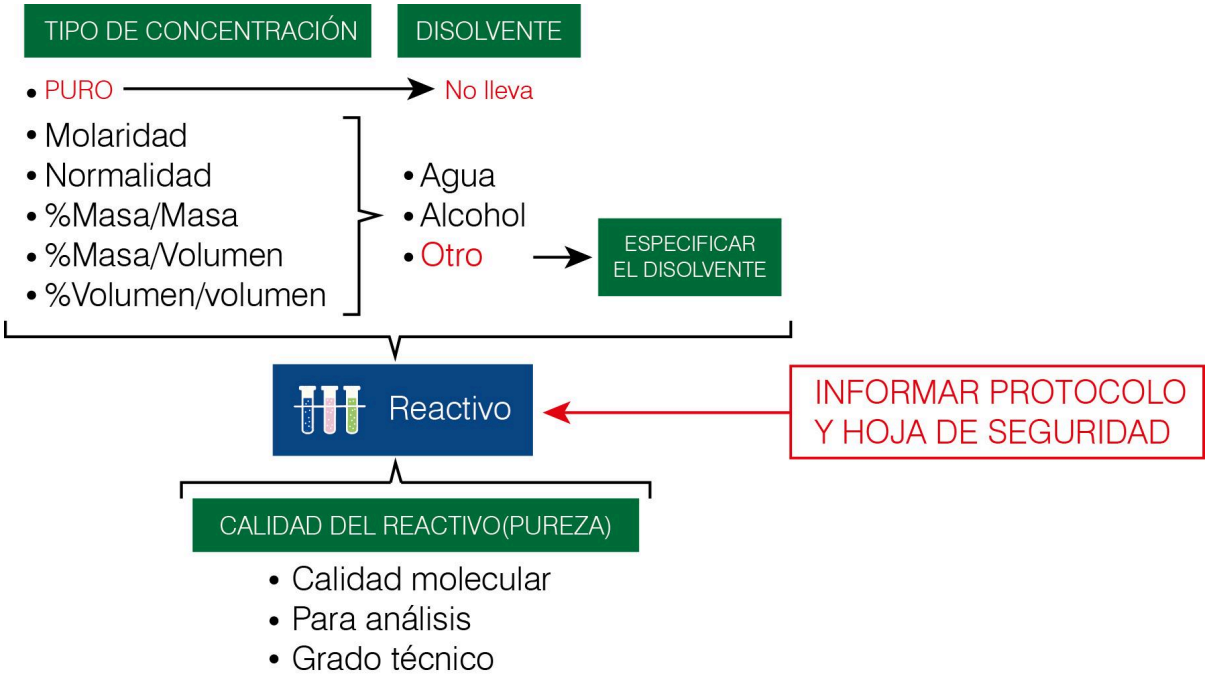
Volver

Reactivos: compuestos químicos añadidos a un sistema para desencadenar una reacción química o para verificar si dicha reacción ocurre. Normalmente se encuentran en mezclas de dos o más sustancias.

Los reactivos pueden ser:

- Puros: Compuestos únicamente por el reactivo en cuestión.
- Mezclas (o disoluciones): Compuestas por el reactivo y un solvente, generalmente agua, aunque también puede ser alcohol u otro solvente especificado en la solicitud. Estas disoluciones tienen un tipo y una medida de concentración.

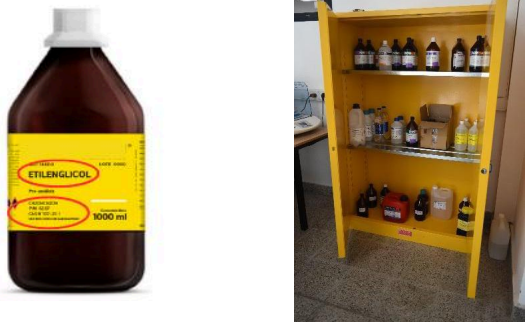
A continuación se presenta un esquema con sus características principales



Al considerar lo anterior, podemos notar la complejidad de entender estos elementos, además de la cuidadosa preparación que requieren por parte del personal del laboratorio. Una manipulación incorrecta puede resultar en accidentes evitables.

Para solucionar este inconveniente, los docentes envían el protocolo de preparación de estas mezclas con todas las aclaraciones correspondientes y la hoja de seguridad de cada reactivo incluido en el pedido.

Volver



Reactivo con su descripción y CAS(Chemical Abstract Service) correspondiente y reactivos inflamables en su armario.

Asimismo, la solicitud debe contar con un encabezado que incluya todos los campos administrativos necesarios para su gestión.

Pedido compuesto por reactivos y materiales preparado por el personal del laboratorio

Tras realizar la solicitud, se necesita un medio de comunicación directa, como un chat, que permita al docente y al personal del laboratorio intercambiar información sobre la solicitud en cuestión.

Volver

Alcance del proyecto

El sistema hasta el momento contaba con el perfil de Docente y el Perfil de laboratorio. Lo que nosotros decidimos agregar, para diferenciar bien los roles, fue un perfil de Admin, para que este sea el único que pueda gestionar el Alta, Baja y Modificación de Usuarios.

El perfil "Personal de Laboratorio", tiene las siguientes características:

Puede ver todos los pedidos realizados por los docentes con la opción de poder ordenarlos por fecha, edificio y estado. En esta actualización se introduce la posibilidad de filtrar pedidos por inactivos/activos.

El sistema les permite visualizar en detalle cada pedido y asignar un edificio y aula para su utilización. Nosotros agregamos la funcionalidad para poder comunicarse por medio de un chat con el docente que realizó el pedido y también poder descargar en PDF el detalle del pedido.

La entrega previa introdujo la gestión de los recursos del laboratorio con la posibilidad de dar de alta un elemento, modificarlo o darlo de baja. En esta actualización, se le quita la capacidad de gestión de usuarios, para que esta facultad sólo pueda llevarla a cabo el Admin del sistema.

Nuestro equipo agrega el perfil "Admin", que cuenta con las mismas funcionalidades del perfil "Personal de Laboratorio", y además con la posibilidad de administrar los usuarios de la aplicación, para generar altas, bajas y modificaciones de los mismos.

El perfil "Docente" puede generar pedidos, por medio de una función, que le permite, seleccionar una fecha de utilización, cargar diferentes tipos de materiales, equipos o reactivos, agregar observaciones, descripciones y visualizar el pedido completo antes de crearlo. En esta entrega, se crea una versión actualizada de la carga de pedidos, con la misma función, pero con mejoras visuales y manejo de errores que facilitan la experiencia de usuario durante la carga de pedidos.

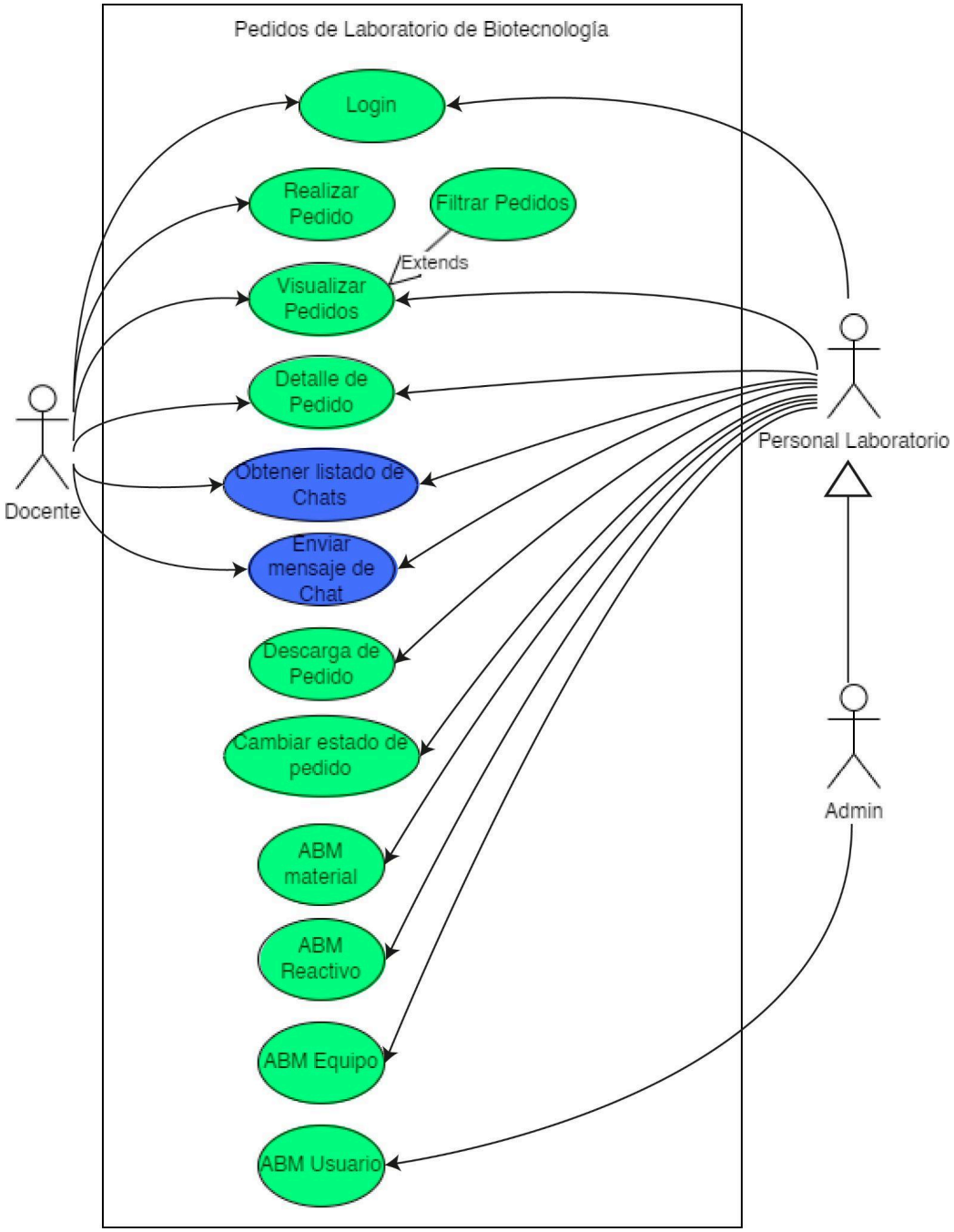
Por medio de la interfaz puede ver todos los pedidos realizados y filtrarlos por fecha o edificio. En esta actualización se introduce la posibilidad de filtrar pedidos por inactivos/activos.

Cada pedido puede ser visualizado en detalle, y también puede comunicarse con el laboratorio por medio de un chat único por pedido realizado.

Diagrama de casos de uso

El diagrama de casos de uso se realizó para visualizar las funcionalidades del sistema. Aquí se tiene una representación más visual y global del mismo con los roles correspondientes, así como también la preponderancia de interacción del laboratorio con el sistema.

En el siguiente diagrama se visualiza las funcionalidades del sistema donde se ve con color azul aquellas nuevas agregadas al proyecto y en verde aquellas que fueron mejoradas o modificadas del proyecto heredado.



Volver

Requerimientos Funcionales

Requerimientos comunes todos los perfiles

RF#1:

Especificación: Login
Introducción: Este requerimiento funcional permite a un usuario ingresar a la aplicación, por medio de sus credenciales.
Entradas:
- Datos de usuario (mínimo 6 caracteres).
- Datos de contraseña (mínimo 6 caracteres).

Procesos:
- Se verifica si el usuario existe en la BD de Usuarios.
  - Si no existe, no se procede con el login.
- Se verifica si la contraseña es correcta con ese Usuario.
  - Si es incorrecta, no se procede con el login.

Salidas:
- Mensaje de confirmación de login exitoso.
- Mensaje de error si las credenciales no son válidas en los inputs.

RF#2:

Especificación: Obtener listado de Chats.
Introducción: Este requerimiento funcional permite a cualquier perfil obtener el listado de chats por id de pedido.
Entradas:
- Datos de Pedido (id)

Procesos:
- Se obtiene listado de mensajes del pedido.
- Se actualizan estados de lectura en los mensajes.

Salidas:
- Listado de los mensajes del chat.

RF#3:

Especificación: Enviar mensaje de chat.
Introducción: Este requerimiento funcional permite a los usuarios poder enviar un chat.
Entradas:
- Dato de Pedido (id).
- Datos de mensaje (string).
- Datos de Usuario (id, dni).

Procesos:
- Se verifica que usuario exista en la BD de Usuarios.
- Se verifica que el mensaje no esté vacío.
- Se envía mensaje y se guarda en BD Mails.

Salidas:
- Mensaje nuevo en pantalla.

Volver

RF#4:

Especificación: Descarga de pedido.
Introducción: Este requerimiento funcional permite al usuario descargar la información. del pedido en formato PDF.
Entradas:
- Dato de Pedido (id)

Procesos:
- Se descarga información del pedido.

Salidas:
- PDF descargado se guarda en la computadora del usuario.

RF#5:

Especificación: Filtrar pedidos.
Introducción: Este requerimiento funcional permite al usuario filtrar los pedidos que tenga disponibles para visualizar.
Entradas:
- Datos de los filtros a aplicar:
  - fecha.
  - edificio.
  - activo/inactivo.

Procesos:
- Se filtran los pedidos por medio de las entradas seleccionadas.

Salidas:
- Listado de los pedidos filtrados con las condiciones indicadas por el usuario.

RF#6:

Especificación: Detalle de pedido.
Introducción: Este requerimiento funcional permite al usuario observar los detalles completos de un pedido.
Entradas:
- Datos de Pedido (id).

Procesos:
- Se verifica que el id sea válido.

Salidas:
- Muestra modal de detalle del pedido.

RF#7:

Especificación: Visualizar pedidos.
Introducción: Este requerimiento funcional permite visualizar todos los pedidos activos hechos por docentes.
Entradas:
- Dato de Usuario (id).

Procesos:
- Se valida que el id de usuario exista.

Salidas:
- Listado de los pedidos hechos por los docentes.

Volver

Perfil Docente

RF#8:

Especificación: Realizar pedido.
Introducción: Este requerimiento funcional permite al docente poder cargar un nuevo pedido.
Entradas:

- Datos Fecha.
- Datos Hora.
- Datos Cantidad de Alumnos.
- Datos Grupos.
- Datos de Observación.
- Datos de Descripción.
- Datos de los materiales a usar.
- Datos de los reactivos a usar.
- Datos de los equipos a usar.
- Datos de la materia.

Procesos:

- Se valida que la fecha no sea menor a la actual.
- Se valida que la cantidad sea mayor o igual a la de grupos.
- Se valida que la hora esté comprendida entre las 6 y 22 hrs.
- Se valida que el campo Observación, Descripción y Materia no estén vacíos.
- Se valida que el material, reactivo o equipo que se agrega exista.
- Se valida que los materiales, reactivos y equipos, se pida una cantidad mayor a la que se denota en stock.
- Se guarda pedido.

Salidas:

- Mensaje de creación de pedido.
- Pedido guardado en BD Pedido.

Perfil Laboratorio

RF#9:

Especificación: Cambiar estado de pedido.
Introducción: Este requerimiento funcional permite cambiar el estado de un pedido a RECHAZADO, ACEPTADO, PENDIENTE.
Entradas:

- Datos de Pedido (id).
- Dato de estado.

Procesos:

- Se valida que el pedido exista en la BD Pedidos.
- Se actualiza la BD Pedido.

Salidas:

- Muestra tarjeta de pedido con flag de color segun el estado nuevo.
- Actualiza la BD de Pedido.

Volver

A continuación de detallan la Alta, Baja y Modificación para el Perfil Laboratorio, o por sus siglas ABM, como se podía apreciar en el diagrama de casos de usos, por eso en este caso se pueden apreciar las funcionalidades especificadas por separado, donde por ejemplo el ABM Material, representa a 3 requisitos funcionales: Alta material, Baja Material y Modificar material.

RF#10:

Especificación: Alta material.
Introducción: Este requerimiento funcional permite crear un material.
Entradas:
- Datos del Material (nombre, clase, stock, enReparacion, activo).

Procesos:
- Se valida que el material no exista en BD Material.
  - Si existe, no se crea el material.
  - Si no existe, se crea el material.
- Se guarda Material.

Salidas:
- Mensaje de creación del material con éxito.
- Material guardado en BD Material.

RF#11:

Especificación: Baja material.
Introducción: Este requerimiento funcional permite dar de baja un material.
Entradas:
- Dato del Material (id).

Procesos:
- Se valida que el material exista en la BD material.
  - si existe, dar de baja el material.
  - si no existe, no dar de baja el material.
- Se da de baja el material.

Salidas:
- Mensaje de baja del material con éxito.
- Base de datos de Material actualizada.

RF#12:

Especificación: Modificar material.
Introducción: Este requerimiento funcional permite modificar un material.
Entradas:
- Datos del Material (id, nombre, clase, stock, enReparacion, activo).

Procesos:
- Verificar que el material exista en la BD Material.
- Se actualiza Material.

Salidas:
- Mensaje de modificación del material con éxito.
- Material guardado en BD Material.

Volver

RF#13:

Especificación: Alta reactivo.
Introducción: Este requerimiento funcional permite crear un reactivo.
Entradas:
- Datos del Reactivo (nombre, CAS, stock, enReparacion, activo).

Procesos:
- Se valida que el reactivo no exista en BD Reactivo.
  - Si existe, no se crea el reactivo.
  - Si no existe, se crea el reactivo.
- Se guarda Reactivo.

Salidas:
- Mensaje de creación del reactivo con éxito.
- Material guardado en BD Reactivo.

RF#14:

Especificación: Baja reactivo.
Introducción: Este requerimiento funcional permite dar de baja un reactivo.
Entradas:
- Dato de Reactivo (id).

Procesos:
- Se valida que el reactivo exista en la BD Reactivo.
  - si existe, dar de baja el reactivo.
  - si no existe, no dar de baja el reactivo.
- Se da de baja el reactivo.

Salidas:
- Mensaje de baja del reactivo con éxito.
- Base de datos de Reactivo actualizada.

RF#15:

Especificación: Modificar reactivo.
Introducción: Este requerimiento funcional permite modificar un reactivo.
Entradas:
- Datos del Reactivo (id, nombre, CAS, stock, enReparacion, activo).

Procesos:
- Se valida que el reactivo exista en la BD Reactivo.
- Se actualiza Reactivo.

Salidas:
- Mensaje de modificación del reactivo con éxito.
- Material guardado en BD Reactivo.

Volver

RF#16:

Especificación: Alta equipo.
Introducción: Este requerimiento funcional permite crear un equipo.
Entradas:
- Datos del Equipo (nombre, clase, stock, enReparacion, activo).

Procesos:
- Se valida que el equipo no exista en BD Equipo.
  - Si existe, no se crea el equipo.
  - Si no existe, se crea el equipo.
- Se guarda equipo.

Salidas:
- Mensaje de creación del equipo con éxito.
- Material guardado en BD Equipo.

RF#17:

Especificación: Baja equipo.
Introducción: Este requerimiento funcional permite dar de baja un equipo.
Entradas:
- Dato del Equipo (id).

Procesos:
- Verificar que el equipo exista en la BD Equipo.
  - si existe, dar de baja el equipo.
  - si no existe, no dar de baja el equipo.
- Se da de baja el equipo.

Salidas:
- Mensaje de baja del equipo con éxito.
- Base de datos de Equipo actualizada.

RF#18:

Especificación: Modificar equipo.
Introducción: Este requerimiento funcional permite modificar un equipo.
Entradas:
- Datos del Equipo (id, nombre, clase, stock, enReparacion, activo).

Procesos:
- Se valida que el equipo exista en la BD Equipo
- Se actualiza equipo.

Salidas:
- Mensaje de modificación del equipo con éxito.
- Material guardado en BD Equipo.

Perfil Administrador

Se detallan Alta, Baja y Modificación para el Perfil Administrador, o por sus siglas ABM, como se podía apreciar en el diagrama de casos de usos.

RF#19:

Especificación: Alta usuario.
Introducción: Este requerimiento funcional permite crear un usuario.
Entradas:
- Datos del Usuario (nombre, apellido, dni, matricula, rol, email).

Procesos:
- Verificar que el usuario no exista en BD Usuario.
  - Si existe, no se crea el usuario.
  - Si no existe, se crea el usuario.

Salidas:
- Mensaje de creación del usuario con éxito.

RF#20:

Especificación: Baja usuario.
Introducción: Este requerimiento funcional permite dar de baja un usuario.
Entradas:
- Datos del Usuario (id).

Procesos:
- Verificar que el usuario exista en BD Usuario.
  - Si existe, se elimina el usuario.
  - Si no existe, no se da de baja el usuario.

Salidas:
- Mensaje de baja del usuario con éxito.

RF#21:

Especificación: Modificar usuario.
Introducción: Este requerimiento funcional permite modificar un usuario.
Entradas:
- Datos del Usuario (id, nombre, apellido, dni, matricula, rol, email).

Procesos:
- Verificar que el usuario exista en la BD Usuario
- Actualizar la BD Usuario.

Salidas:
- Mensaje de modificación del usuario con éxito.

Requerimientos funcionales		
Modificado		
Nombre	RF	Modificaciones
Login	#1	Se añadieron verificaciones, errores descriptivos. El código se refactoriza, para ser más compacto y escalable, valiéndose de la librería React-hook-Form, que reduce la cantidad de hooks de estado, lo que hace que el código sea más legible. También se cambia la forma de la llamada al servicio, para que envíe la contraseña encriptada y el usuario se envíe por medio del body de Axios y no por url.
Filtro de pedidos	#5	Se agregó el filtro de “incluir inactivos” ya que por defecto no discriminaba los pedidos vencidos.
Ver detalle de pedido	#6	Se rediseñó visualmente, con una vista más intuitiva y agradable, con la posibilidad de acceder a la nueva modalidad de chat y también descarga de pedidos.
Visualizar pedidos	#7	Se agregó paginación al llamado del servicio para poder traer los pedidos por tandas. Para optimizar el tiempo de carga del navegador, y no tener que renderizar todos los pedidos que existen en la Base de datos, lo que podría ralentizar drásticamente la carga de la aplicación web.
Realizar pedido	#8	Se agregó el stepper que permite la carga del pedido por medio de pantallas con pasos consecutivos, con una vista previa del pedido antes de crearlo. Se modificaron las validaciones. Se agregó control de stock y un sistema de turnos para la reserva de los materiales.
Alta-Modificar reactivos/materiales/equipos	#10 #12 #13 #15 #16 #18	Se agregó el borrado lógico que es una herramienta que permite a los administradores del laboratorio poder gestionar de forma más eficiente cuáles materiales, reactivos y equipos se encuentran disponibles durante la carga de pedidos, impide que se carguen materiales que figuren como no disponibles pero mantiene la integridad en la base de datos. También se agregó la opción de gestión de los recursos en reparación, que estarían de baja de forma temporal.
Alta/Modificar/ Baja Usuario	#19 #20 #21	Se le agregó la función de que solamente el Perfil Admin pueda visualizar, crear, modificar y eliminar a los usuarios. También se modificó para que visualmente el Perfil Admin no se pueda eliminar a si mismo.

Volver

Nuevos		
Nombre	RF	Descripción
Obtener listados de Chats	#2	Los usuarios pueden obtener el historial de conversaciones realizadas de un pedido determinado.
Enviar un chat	#3	Los perfiles de usuarios y el laboratorio pueden mantener una conversación en un chat de forma sincrónica por medio del WebSocket. Esto se hace posible al abrir una sesión de comunicación interactiva entre el navegador del usuario y un servidor (protocolo HTTP) que realiza una conexión dúplex para comunicarse.
Descarga de pedido	#4	Los perfiles de usuarios y el laboratorio pueden descargar el detalle del pedido en formato PDF.

Requerimientos no funcionales

Los requerimientos no funcionales son aquellos que definen las características de un sistema o software que no están directamente relacionadas con la funcionalidad del mismo, sino con aspectos como el rendimiento, la seguridad, la usabilidad, la escalabilidad, el rendimiento, entre otros.

Estos requerimientos son igualmente importantes que los requerimientos funcionales, ya que garantizan que el sistema cumpla con ciertos estándares de calidad y desempeño.

4 requerimientos no funcionales para “Pedidos de Laboratorio de Biotecnología”:

- Seguridad:
    - Revisar el proceso de envío de contraseñas convertidas a base64 antes de ser enviadas al backend, ya que esto puede comprometer la seguridad de las credenciales de los usuarios, luego reconvertirla y hashear para almacenar en Base de datos con Bcrypt.js con un salt mínimo de 10. Se recomienda enviar las contraseñas de forma segura y encriptada para evitar posibles vulnerabilidades.
    - Asegurar que la información sensible almacenada en el frontend, como el rol, nombre, apellido, DNI, email y matrícula del usuario en el localstorage, esté debidamente protegida y hasheada con Crypto.js (AES) con un salt mínimo de 16 para mantener la confidencialidad de los datos.
    - Se implementará un control de acceso basado en roles para garantizar que solo los administradores puedan realizar acciones en la sección de ABM de usuarios.

Volver

- Se utilizarán JSON Web Tokens para establecer un mecanismo seguro de autenticación entre el frontend y el backend del sistema.
- Se implementará una protección CORS (Cross-Origin Resource Sharing) para restringir y controlar los accesos al backend desde dominios externos.

- Usabilidad:

  - Mantener una interfaz de usuario intuitiva y fácil de usar, gracias a las capacidades de React y Material-UI para crear un diseño atractivo y funcional.
  - Proporcionar instrucciones claras y ayuda contextual utilizando las funcionalidades de Material-UI para generar mensajes informativos y guiar a los usuarios en caso de dudas.
  - Realizar pruebas de usabilidad con docentes reales para identificar posibles puntos de mejora en la interacción del sistema y así garantizar una experiencia de usuario óptima.

- Rendimiento:

  - Optimizar el rendimiento del sistema a nivel de backend, debido a que las consultas a MongoDB Atlas estén indexadas adecuadamente y que no haya operaciones costosas que afecten el rendimiento.
  - Utilizar técnicas de optimización de consultas en MongoDB, como la optimización de índices y la proyección selectiva, para garantizar respuestas rápidas a las consultas realizadas desde el frontend.
  - Implementar estrategias de carga progresiva en el frontend para mejorar la velocidad de carga de la interfaz y optimizar el rendimiento general del sistema.

- Escalabilidad:

  - Diseñar una arquitectura modular y escalable que permita agregar nuevos consorcios y propietarios sin afectar el rendimiento del sistema.
  - Utilizar tecnologías y plataformas escalables en la infraestructura, como servicios en la nube, para adaptarse a un mayor volumen de datos y transacciones.
  - Monitorear constantemente el rendimiento del sistema y realizar ajustes en la capacidad de acuerdo con las necesidades del crecimiento.

Volver

Metodología de trabajo

Para la realización de este proyecto se utilizó una adaptación de la metodología **Scrum** que es un proceso que se aplica de manera regular un conjunto de buenas prácticas para **trabajar colaborativamente,** en **equipo**, y obtener el mejor resultado posible de un proyecto. Esto incluye una fase de planificación llamada **Sprint Planning**, donde se define el trabajo a realizar durante un período breve de tiempo fijo denominado **sprint**. Durante cada sprint, el equipo de Scrum trabaja para completar una cantidad de trabajo establecida. Es decir, el proyecto se divide en pequeñas partes para poder abordarlas de forma más rápida y eficiente.

Esquema de Metodología Scrum

Ponemos en práctica esta metodología con el inicio del sprint que corresponde a una duración de tres semanas. Al comienzo del mismo realizamos una ficha de inicio de sprint que está compuesto por:

- Nombre de la ficha de inicio de sprint.
- Nombre del grupo que realiza el proyecto.
- Número del sprint, fecha de inicio y de cierre.
- objetivos que se van a querer implementar en ese sprint.

Esto nos ayuda a tener en cuenta la fecha que durará el sprint y lo que se va a querer implementar.

Volver

Desarrollo de aplicaciones - 2do cuatrimestre 2023

Ficha de principio de sprint

Grupo - proyecto
Grupo 2 - Pedidos de materiales para laboratorio

Nro de sprint	Fecha de comienzo	Fecha de fin
1	17/08/2023	07/09/2023

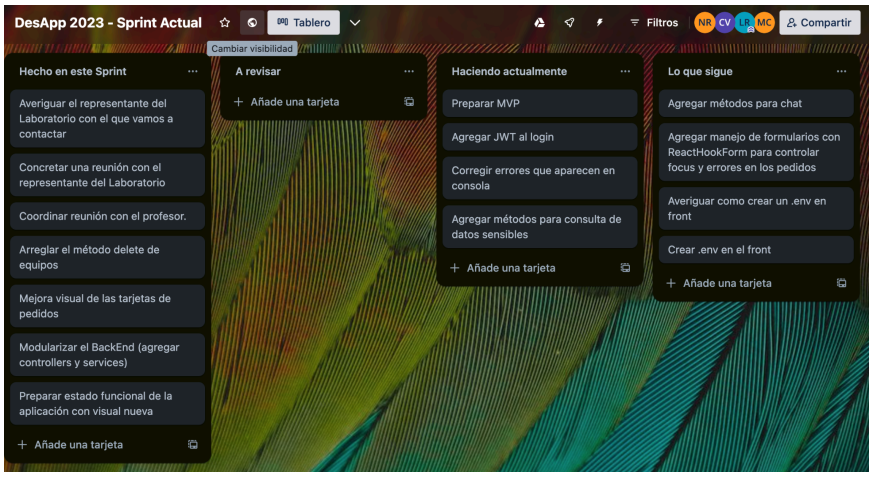
Objetivos que nos ponemos para este sprint (una frase para cada uno).

*Levantar los ambientes (Backend y Frontend)
*Contactar a algún integrante del curso anterior para validar los repositorios.
*Revisar el código para identificar desde dónde partimos y las diferentes tecnologías que se utilizan/profundizar en las mismas.
*Buscar posibles BUGS/Código no funcional/Innecesario o redundante.
*Coordinar reunión con representante/s del laboratorio.
*Empezar a identificar faltantes relacionados con UX/UI (falta de mensajes de error, manejo de errores, problemas visuales, etc.)
*Desarrollar una estructura de separación por módulos (por ejemplo tomar los servicios y separarlos por entidades o por métodos).
*Preparar una estructura de GitFlow partiendo de Main, creando la rama de Development y desde allí crear las ramas necesarias para ir agregando código.

Al realizar la ficha de inicio de sprint, se utiliza Trello que es una herramienta visual que permite a los equipos gestionar cualquier tipo de proyecto y flujo de trabajo, así como supervisar tareas. Para organizar el trabajo separamos en columnas:

- Lo que sigue: Consiste en ingresar todas las tareas que se encuentran en la ficha de inicio de Sprint.
- Haciendo actualmente: En esta columna se ingresan las tareas que están en desarrollo por algún integrante del equipo.
- A revisar: Se ingresan las tarjetas que se van a testear. Si el testeo fue con éxito se mueven a “Hecho en este Sprint” en caso contrario vuelve a “Haciendo actualmente”.
- Hecho en este Sprint: Es la última columna en la cual las tarjetas se mueven si fueron testeadas con éxito.

Volver



Para continuar con la metodología Scrum teníamos **weekly** una vez por semana con el docente para mostrarle cómo veníamos con las tareas del trello. Al final del sprint nos reunimos con el profesor para realizar una review y retrospectiva que consiste mostrarle la ficha de final de sprint que es idéntica a la de inicio pero con la diferencia que tiene una descripción con las tareas finalizadas y las que no se llegaron hacer.

Desarrollo de aplicaciones - 2do cuatrimestre 2023

Ficha de fin de sprint

Grupo - proyecto
Grupo 2 - Pedidos de materiales para laboratorio

Nro de sprint	Fecha de comienzo	Fecha de fin
1	17/08/2023	07/09/2023

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

A qué llegamos, o nos acercamos bastante.

*Levantar los ambientes (Backend y Frontend)
*Contactar a algún integrante del curso anterior para validar los repositorios.
*Buscar posibles BUGS/Código no funcional/Innecesario o redundante.
*Empezar a identificar faltantes relacionados con UX/UI (falta de mensajes de error, manejo de errores, problemas visuales, etc.)

A qué no llegamos.

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

ficha de fin de Sprint

Volver

Para finalizar con la metodología Scrum se muestra cómo quedó la tabla del plan de trabajo que representa los 10 sprint de trabajo y que se hizo en cada respectivo sprint.

	1	2	3	4	5	6	7	8	9	10
Arq										
RF y NF										
Dise										
Front										
Back										
Doc										
inf Final										

plan de trabajo

Prototipo en Figma

Para realizar los mock up de la aplicación y tener una visión general de lo que desarrolla utilizamos Figma. Esta es una herramienta para prototipado web y diseño colaborativo. El uso fundamental que le dimos fue el diseño de la interfaz de usuario. Nos permitió visualizar las pantallas, identificar el flujo de navegación entre las mismas para finalmente realizar una demostración al personal de laboratorio antes de comenzar con el desarrollo de la aplicación.

Al ser un proyecto heredado, tuvimos reuniones donde propusimos cambios visuales modernos y más agradables al actual. Este cambio produjo que el proyecto obtenga una mejor aceptación, y con ello surgieron nuevos requerimientos funcionales para enriquecerlo.

Seguidamente se plantea una comparación del Figma propuesto con las pantallas finales de la aplicación.

Volver

Login

- Heredado

- Propuesta en Figma

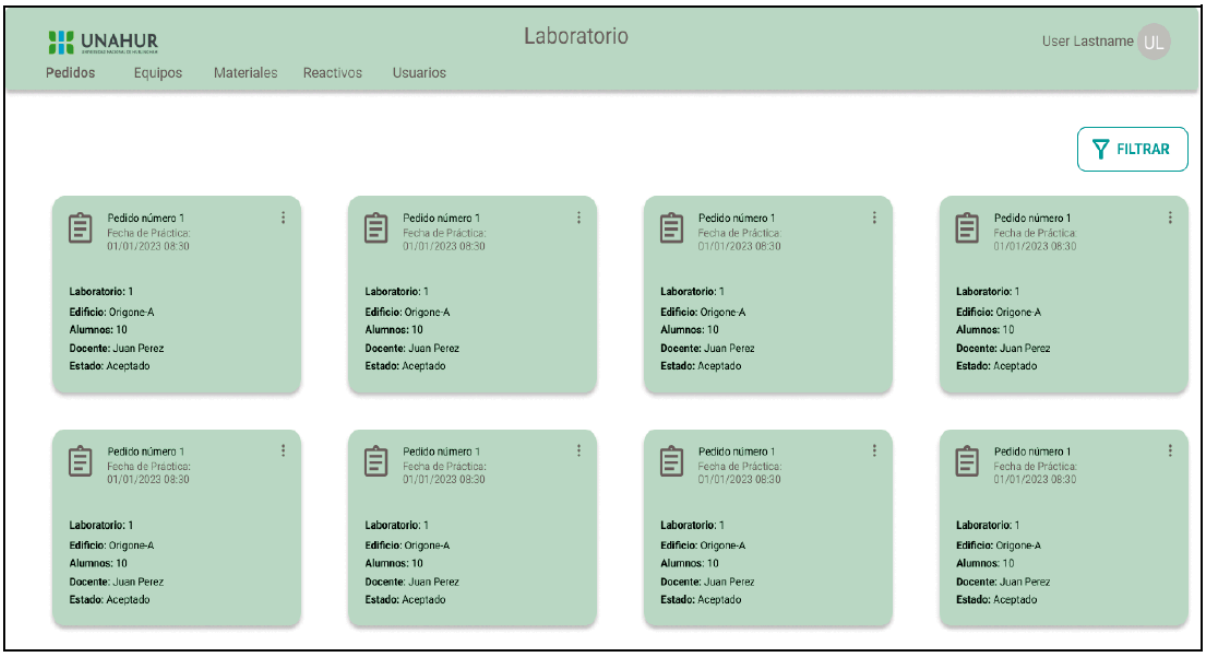
Volver

- En aplicación

Como se puede ver el cambio de la pantalla de login heredado, y la pantalla actual del figma representa un cambio visual con un enfoque explícitamente moderno. Luego si bien la pantalla de Figma y la actual de la aplicación son similares, decidimos retirar la posibilidad de recuperar contraseña, ya que no fue necesaria, porque el encargado de administrar todas a los usuarios es solo el perfil Admin.

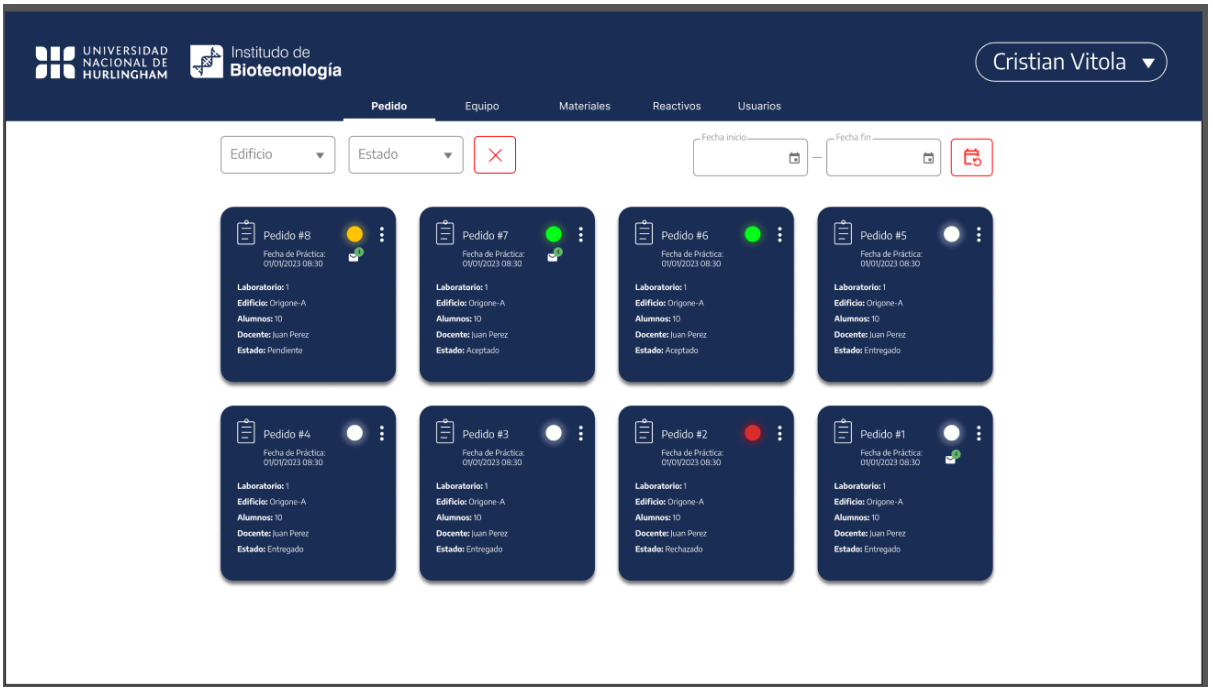
Pantalla de visualización de pedidos

- Heredado



Volver

- Propuesta en Figma



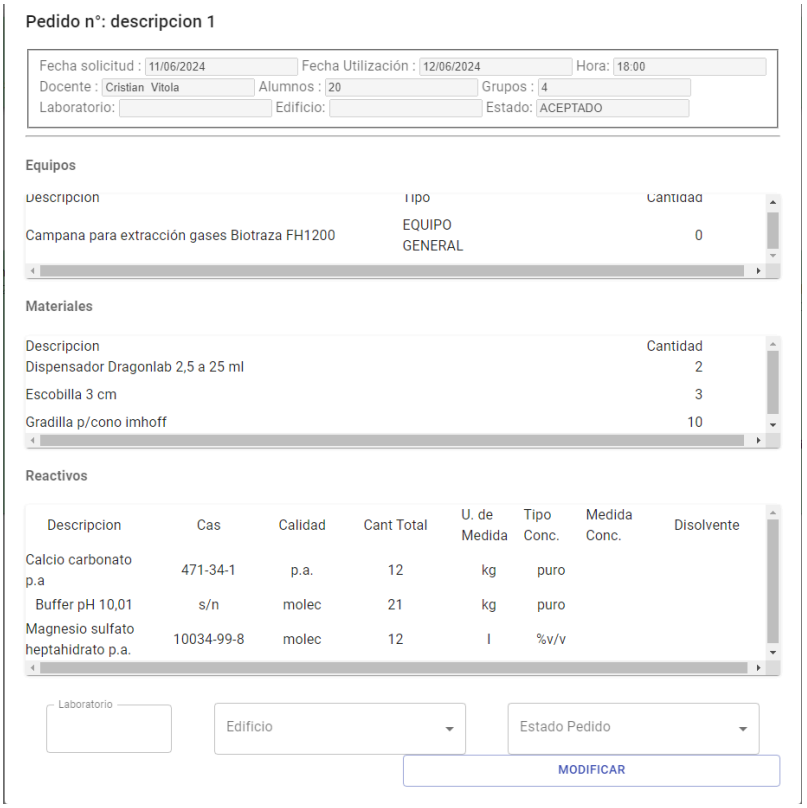
- En aplicación

En este caso ambos, prototipo y pantalla, son semejantes a lo que información se refiere, pero tienen cambios como el agregado del botón de descarga por pedido y la posibilidad de excluir pedidos inactivos, que permiten una mejor experiencia de usuario.

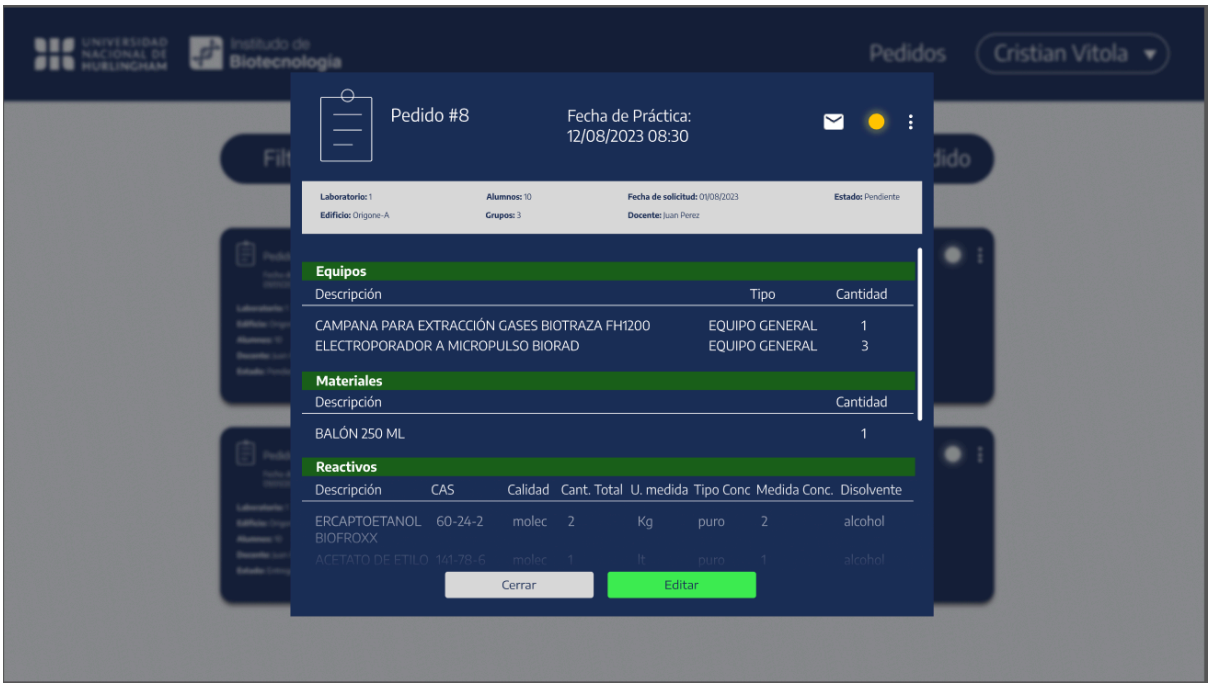
Volver

Pantalla de visualización de Detalle de Pedido

- Heredado

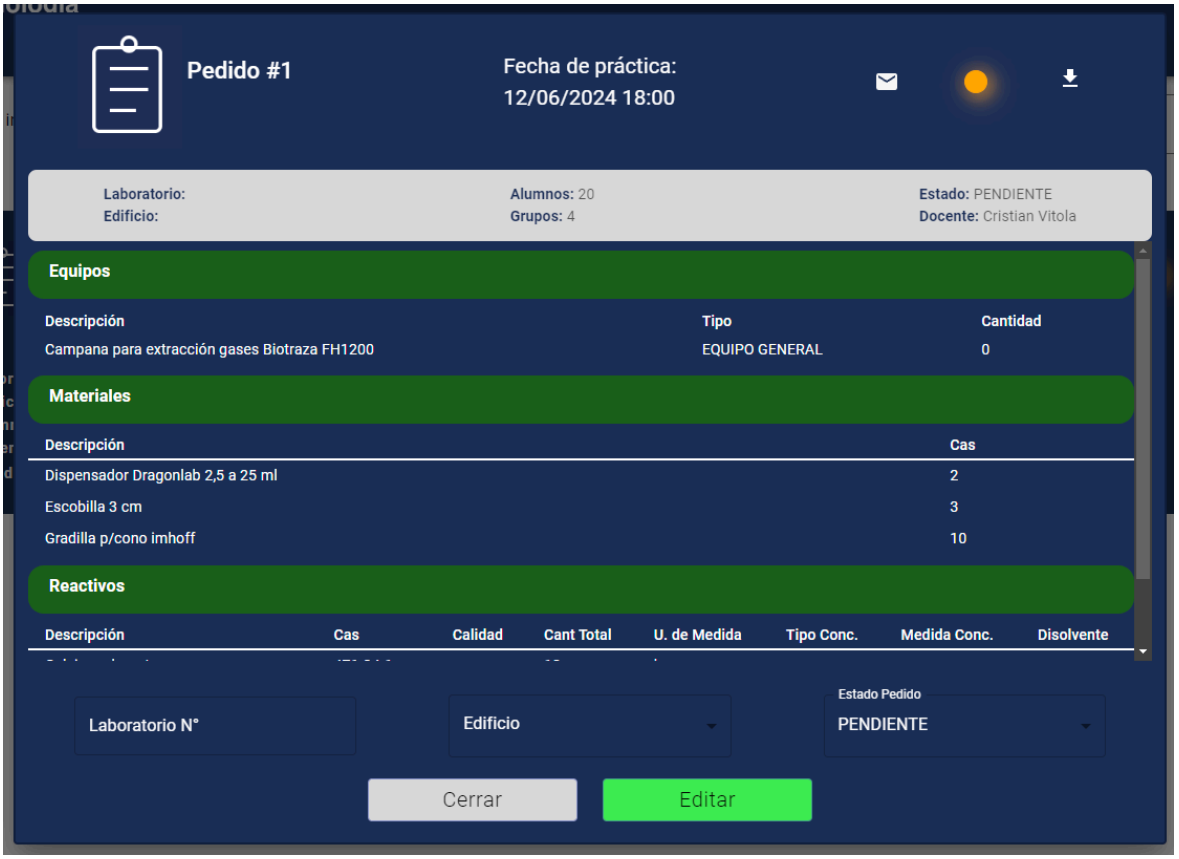


- Propuesta en Figma



Volver

- En aplicación



En este caso ambos, prototipo y pantalla, son semejantes a lo que información se refiere, pero tienen cambios como el agregado del botón de descarga por pedido, también se redondearon las puntas de los modales, por recomendación de los profesores.

Volver

Pantalla de comunicación por chat entre Docentes y Laboratorio

- Propuesta en Figma

- En Aplicación

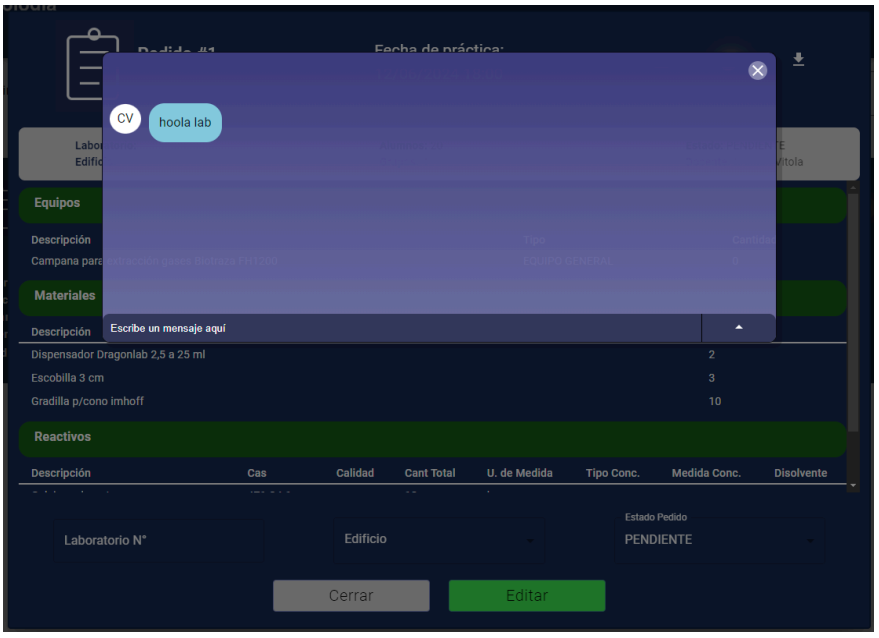


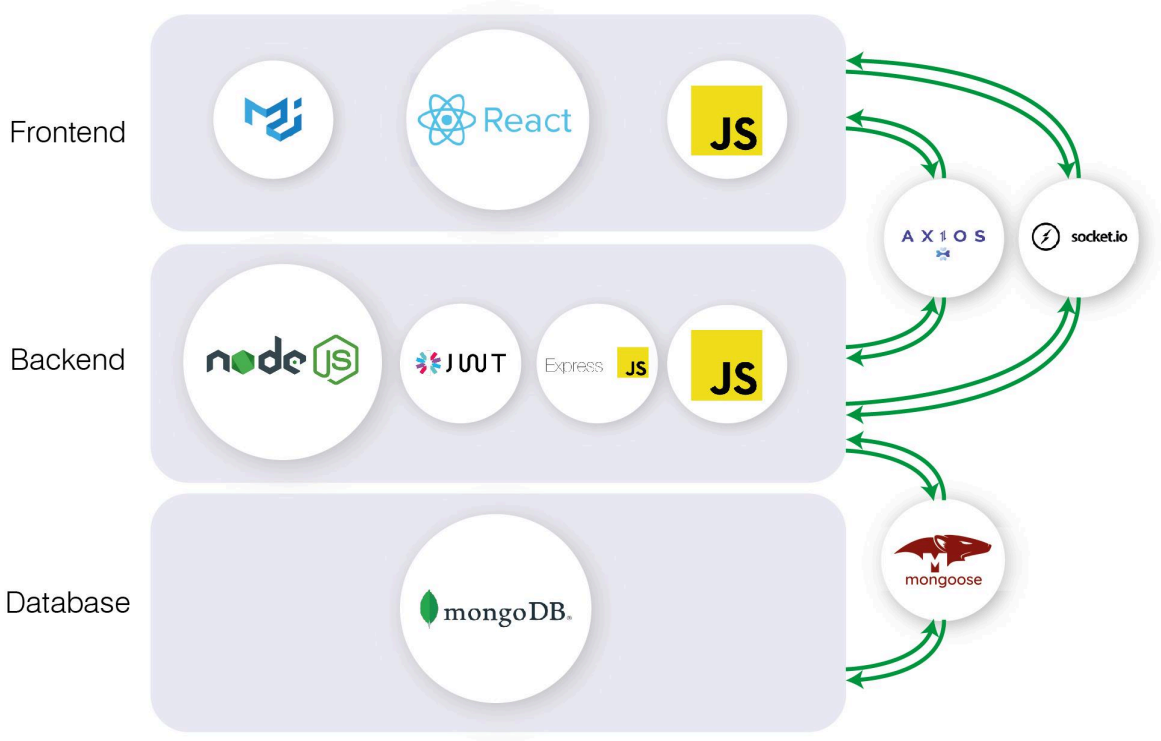
Diagrama de arquitectura

El diagrama de arquitectura en un proyecto de software sirve para visualizar y documentar la estructura de la aplicación, muestra las diferentes capas y componentes que la componen, así como las relaciones entre ellos. Esto facilita la comunicación entre los miembros del equipo de desarrollo, ayuda a planificar y organizar el trabajo de manera eficiente, y permite identificar posibles problemas o áreas de mejora en la arquitectura del

Volver

software. Además, el diagrama de arquitectura es útil para tener una visión general del proyecto y para facilitar su mantenimiento a lo largo del tiempo.

Debido a que se trata de un proyecto heredado, no se pudo elegir las herramientas ni la arquitectura. A continuación se detalla la arquitectura actual del proyecto:



Arquitectura del proyecto

- **MongoDB**: Es una base de datos NoSQL orientada a documentos. Cada registro es un documento que consta de pares clave-valor, similares a los objetos JSON (JavaScript Object Notation). MongoDB es flexible y permite a los usuarios crear esquemas, bases de datos y colecciones sin los requerimientos de una base de datos SQL tradicional.
- **Express.js**: Es un framework que se ejecuta sobre Node.js y se utiliza para desarrollar el backend de aplicaciones web. Node.js está diseñado para ejecutar JavaScript en el servidor, pero no está optimizado para desarrollar sitios web por sí solo. Express.js complementa a Node.js y le proporciona una estructura y funciones específicas para la creación de sitios web.
- **React.js**: Es una biblioteca creada por Facebook, ampliamente utilizada para desarrollar componentes de interfaz de usuario en aplicaciones web de una sola página (SPA). React.js facilita la creación de interfaces de usuario atractivas y dinámicas.
- **Node.js**: Es un entorno de ejecución para JavaScript que permite ejecutar código JavaScript en el servidor, fuera del navegador. Node.js utiliza un sistema de módulos, que permite organizar y reutilizar el código de manera eficiente. Cada módulo tiene su propio contexto, lo que evita interferencias entre ellos y permite desarrollar proyectos de manera modular y predecible.

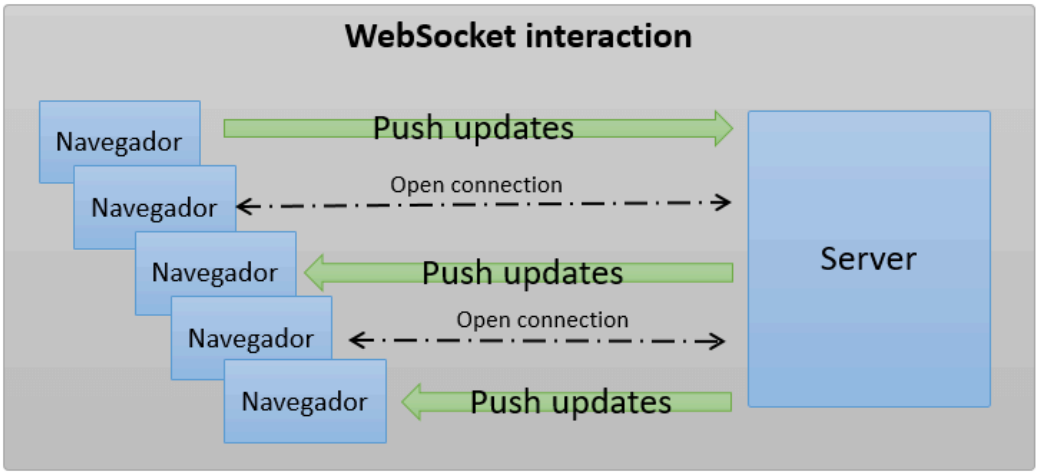
Volver

- **Mongoose**: Es una biblioteca de modelado de objetos (Object Data Modeling, ODM) diseñada para MongoDB y Node.js. En el contexto de aplicaciones Node.js que utilizan MongoDB como base de datos, Mongoose proporciona una capa de abstracción sobre MongoDB que simplifica la interacción y la gestión de datos.

El stack agregado por nosotros es el siguiente:

- **Socket.io**: Es una librería *open source* de **JavaScript** basada en el protocolo Websocket para **Node.js** que permite una comunicación TCP bidireccional en tiempo real entre clientes y servidor. Para ello se basa principalmente en *Websocket* pero también puede usar otras alternativas como *sockets* de Adobe Flash, JSONP polling o long polling en AJAX.

  Está implementación en el proyecto mejora la experiencia del usuario al proporcionar una comunicación fluida y en tiempo real. Además es altamente escalable y robusto, lo que asegura que nuestro proyecto pueda manejar múltiples conexiones simultáneas de manera eficiente y confiable.



arquitectura webSocket

- **JWT**: Es un estándar abierto (RFC 7519) que define un método compacto y autocontenido para la transmisión segura de información entre partes codificadas como un objeto JSON. Esta información puede verificarse y ser fiable porque está firmada digitalmente. Los JWT se firman con un secreto (con el algoritmo HMAC) o un par de claves públicas/privadas que utilizan RSA.

  La integración de JWT en nuestro proyecto ofrece varios beneficios como autenticar y verificar la identidad del usuario de manera segura, lo que es crucial para la protección de datos sensibles y el acceso seguro a recursos. Además de ser una mejora en el proyecto decidimos mantener las buenas prácticas de desarrollo aprendidas en el transcurso de la carrera.

Volver

Visualización de una como se realiza un login, y como JWT devuelve un valor encriptado de autorización, que luego es utilizado para solicitudes al servidor

A continuación se describen los pasos para realizar una petición al servidor de la aplicación web:

1. El usuario envía una petición al servidor a través de la aplicación web, con esto logra la interacción del frontend con ReactJS.
2. Node.js recibe la petición del cliente, la procesa y maneja.
3. Express.js se encarga de enviar la petición a MongoDB.
4. Una vez que MongoDB responde, Express.js recibe los datos y prepara la respuesta.
5. Node.js recibe la respuesta de Express.js y la envía de vuelta al frontend.
6. Finalmente, ReactJS recibe la respuesta y la presenta al usuario.

Además de agregar nuevas herramientas al stack, se mejoró la arquitectura de carpetas del proyecto backend. El proyecto original que heredamos tenía una carpeta llamada **`models`** y otra llamada **`routes`**. En el archivo **`routes.js`** se encontraban todas las rutas con los servicios.

Volver

arquitectura del proyecto back heredado

Esto hace que el código no cumpla con el conjunto de principios y prácticas destinados a escribir código claro, comprensible y mantenible. Tener un archivo con todas las rutas de los servicios crea un archivo con muchas líneas de código, lo que lo hace muy poco legible y difícil de escalar.

Las mejoras que se realizaron incluyeron la separación de las rutas en varios archivos más pequeños y legibles, lo que facilita su funcionalidad y modulación, utilizamos un concepto que se llama modelo-ruta-controlador. Este modelo facilita la escalabilidad del proyecto a medida que crece y mejora la eficiencia del equipo de desarrollo.

El modelo plantea tener una carpeta **`routes`**, con archivos representativos de cada entidad, como **`auth.route.js`** (referencia a autenticación) dentro de cada uno, en el cual solo se visualizan las peticiones http.

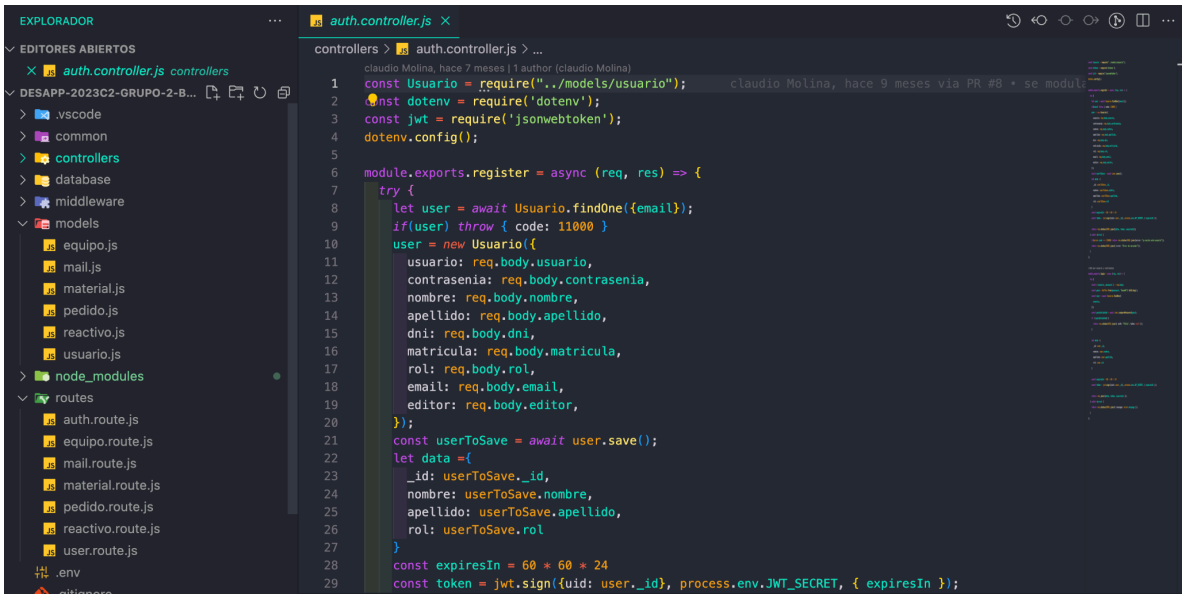
```
router.post("/register",verifyToken ,register);
router.post("/login", login);
```

contenido de auth-route.js, con sus endpoint http

Por otra parte, como se puede ver en la imagen, la lógica de negocio empleada para resolver esa petición, no se realiza en el mismo archivo, para eso se crea una carpeta **`controllers`** y dentro se ponen archivos representativos de la entidad, se usa una convención , **`auth.controller.js`**. Estos archivos si van a contener la lógica de cada petición, y luego van a ser exportados e importados en los archivos de **`routes`**.

La relación entre los archivos de rutas y los archivos de controladores, donde se importan y exportan las funciones, facilita la reutilización y modularidad del código.

Volver



contenido de auth.controller.js, con lógica de negocio

Relaciones entre pantalla y llamadas a servicio

En la siguiente tabla se visualiza las funcionalidades del sistema donde se ve con color naranja aquellas nuevas agregadas al proyecto y aquellas que fueron mejoradas o modificadas en violeta del proyecto heredado.

Requerimiento funcional	Pantalla	Endpoint
Generales		
Login	/login	/api/auth/login(POST)
Obtener listados de Chats	/Laboratorio/Pedidos /Docente/Pedidos	/api/mail/mails/:id(GET)
Enviar un chat	/Laboratorio/Pedidos /Docente/Pedidos	/api/mail/send(POST)
Ver detalle de pedido	/Laboratorio/Pedidos /Docente/Pedidos	/api/pedido/getOne(GET)
Docente		
Realizar pedido	/Docente/pedido	/api/pedido/post(POST)
Visualizar pedidos por dni	/Docente/pedido	/api/pedido/getAllByDni(GET)
Laboratorio		

Volver

Visualizar pedido	/Laboratorio/Pedidos	/api/pedido(GET)
Cambiar estado de un pedido	/Laboratorio/Pedidos	/api/pedido/update/:id(PATCH)
Modificar material	/Laboratorio/Materiales	/api/material/update/:id(POST)
Modificar reactivo	/Laboratorio/Materiales	/api/reactivo/update/:id(POST)
Modificar equipo	/Laboratorio/Materiales	/api/equipo/update/:id(POST)
Crear material	/Laboratorio/Equipos	/api/material/post(POST)
Crear reactivo	/Laboratorio/Equipos	/api/reactivo/post(POST)
Crear equipo	/Laboratorio/Equipos	/api/equipo/post(POST)
Eliminar material	/Laboratorio/Reactivos	/api/material/delete(POST)
Eliminar reactivo	/Laboratorio/Reactivos	/api/reactivo/delete(POST)
Eliminar equipo	/Laboratorio/Reactivos	/api/equipo/delete(POST)
Admin		
Modificar usuario	/Laboratorio/Usuarios	/api/usuario/update/:id(PATCH)
Crear usuario	/Laboratorio/Usuarios	/api/auth/register(POST)
Eliminar usuario	/Laboratorio/Usuarios	/api/usuario/delete/:id(DELETE)

Volver

Diagrama de colecciones de la base de datos

En el diagrama de bases de datos heredado encontramos dos entidades que son **laboratorio** y **docente**. Al aplicar buenas prácticas para modelar la base de datos, se crea una entidad **usuario** que permite centralizar la gestión de usuarios y luego se determinan distintos roles para poder diferenciar cada uno de ellos: laboratorio, docente y admin.

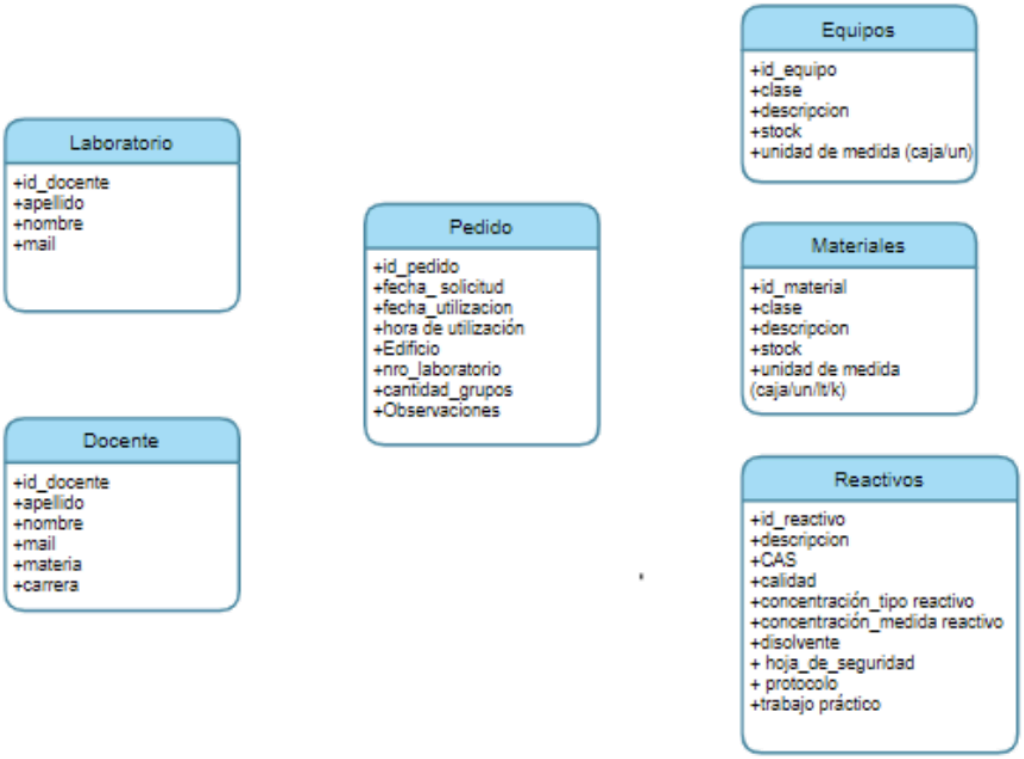


diagrama de bases de datos heredado

En el siguiente diagrama se puede observar que las entidades de Laboratorio y Docente, son unificadas en una sola entidad **Usuario**, diferenciadas por el atributo "*rol*". También se agrega la entidad **Mails**, que guarda los mensajes que se realizan en cada pedido.

Se agregan los atributos "*enUso*", "*enReparacion*" y "*stock*", para administrar la disponibilidad de los **Materiales**, **Equipos** y **Reactivos**.

Se agrega el atributo "*vigente*", en el caso de sí un **Pedido** se encuentra inactivo por tener 10 días de antigüedad a la fecha actual.

Volver

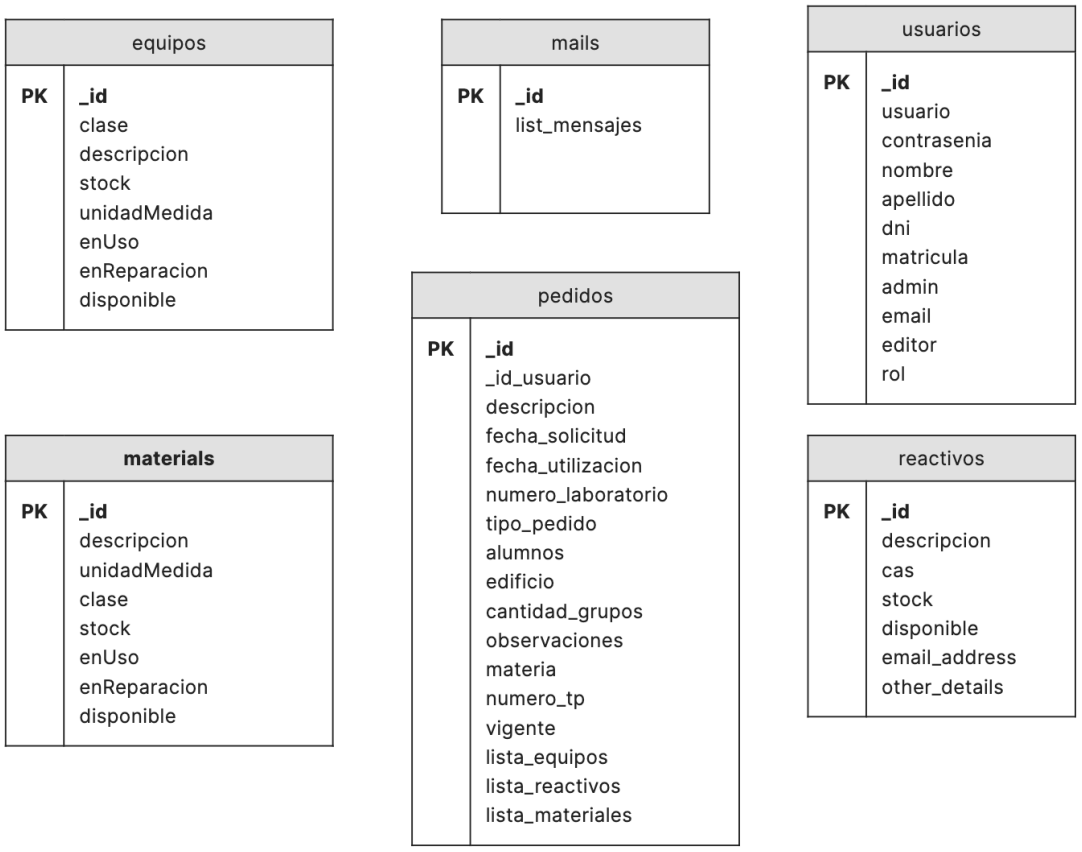


diagrama de base de datos

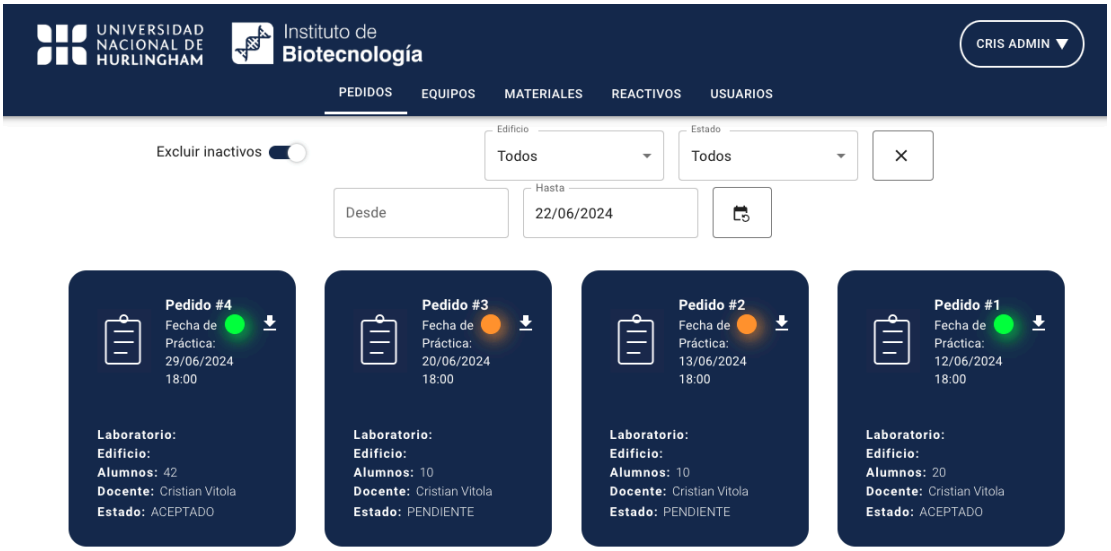
Set de pruebas

Los conjuntos de pruebas son esenciales para validar la funcionalidad correcta de una aplicación. Generalmente se ejecutan a un nivel más alto, y emulan las interacciones del usuario con la aplicación. Esto asegura que las pruebas reflejen de manera precisa la experiencia real del usuario, que facilita la detección de posibles problemas que podrían pasar inadvertidos con otras estrategias. En este contexto, se presentan casos de prueba basados en los requisitos funcionales, con el fin de identificar cualquier defecto que pueda no haber sido contemplado en los requisitos de la aplicación.

Login	
Descripción	Verificar credenciales del usuario para ingresar al sistema.
Condiciones	El usuario debe estar registrado en el sistema.
Entrada	Nombre de usuario y contraseña.
Resultado esperado	Permitir acceder al sistema al usuario.
Evaluación de la prueba	Login exitoso.

Volver

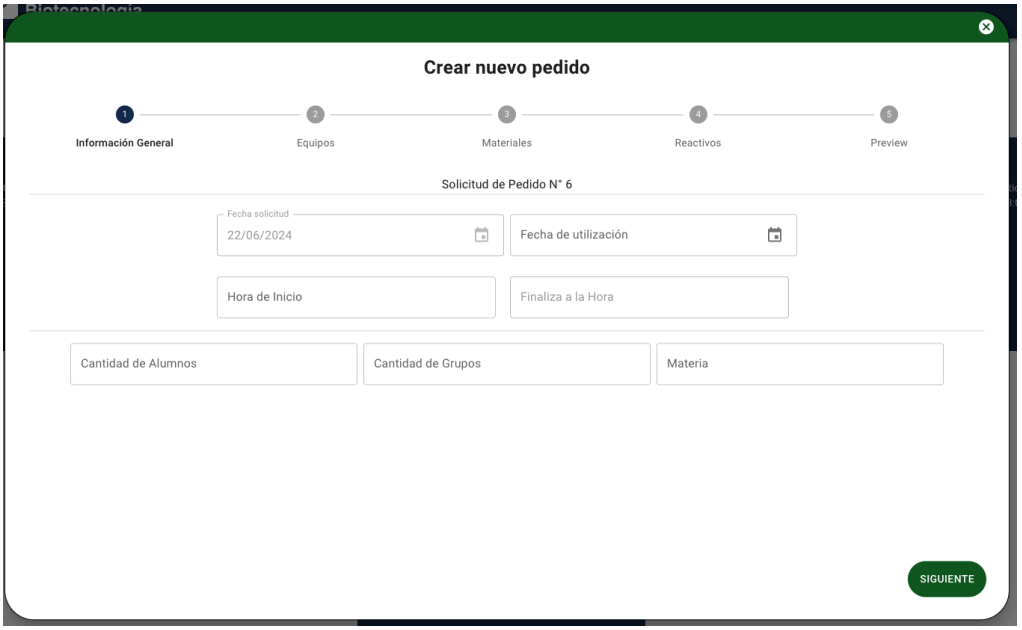
Vista de Login.



Login exitoso.

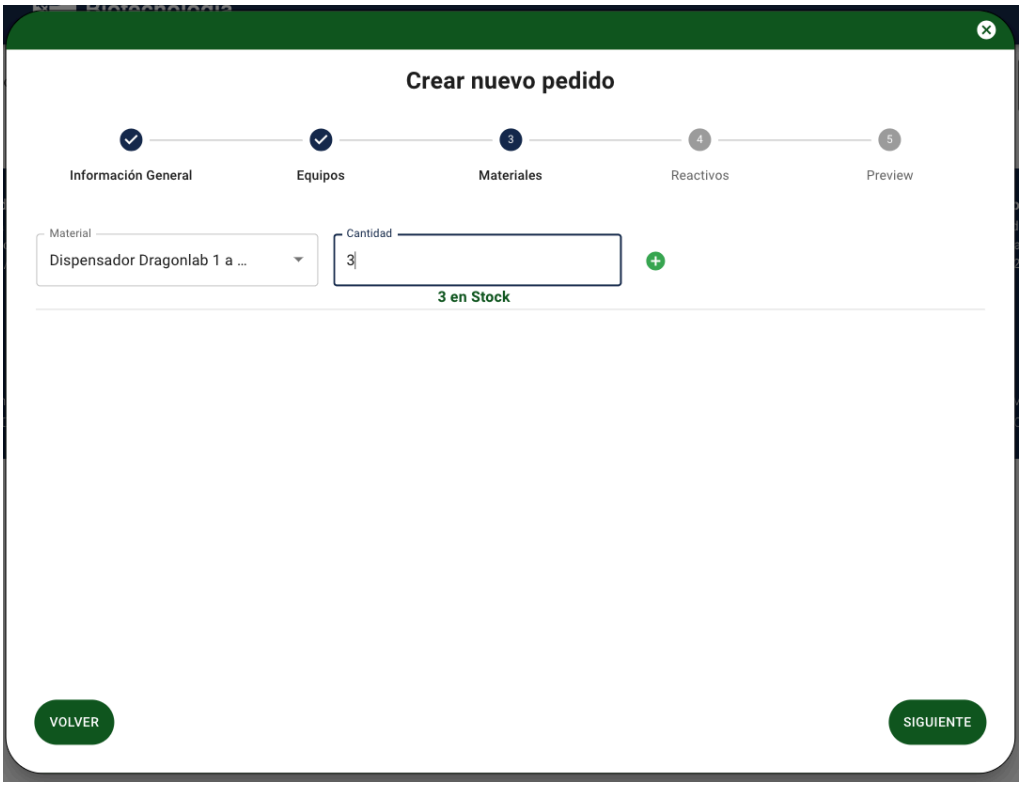
Volver

Realizar pedido	
Descripción	Permitir al docente realizar un nuevo pedido de materiales.
Condiciones	El usuario debe tener perfil docente.
Entrada	• Datos Fecha • Datos Hora • Datos Cantidad de Alumnos • Datos Grupos • Datos de los materiales a usar • Datos de los reactivos a usar • Datos de los equipos a usar • Datos de la materia
Resultado esperado	Permitir crear el pedido.
Evaluación de la prueba	Crea el pedido correctamente.

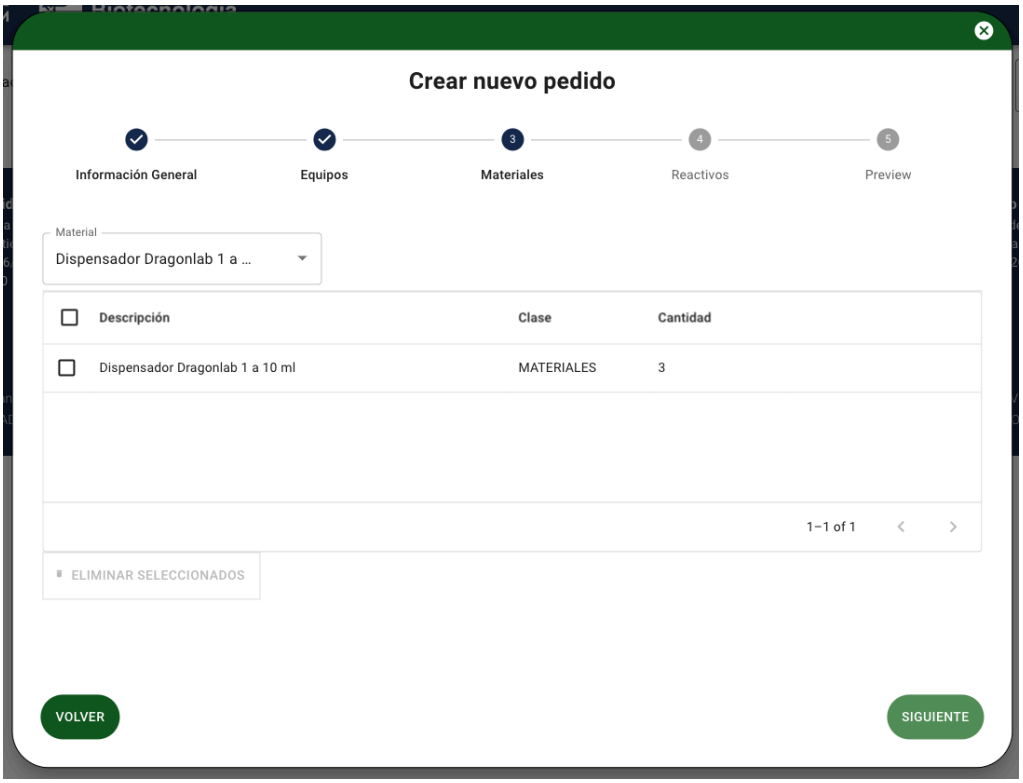


Vista de carga de información general del pedido por pasos.

Volver

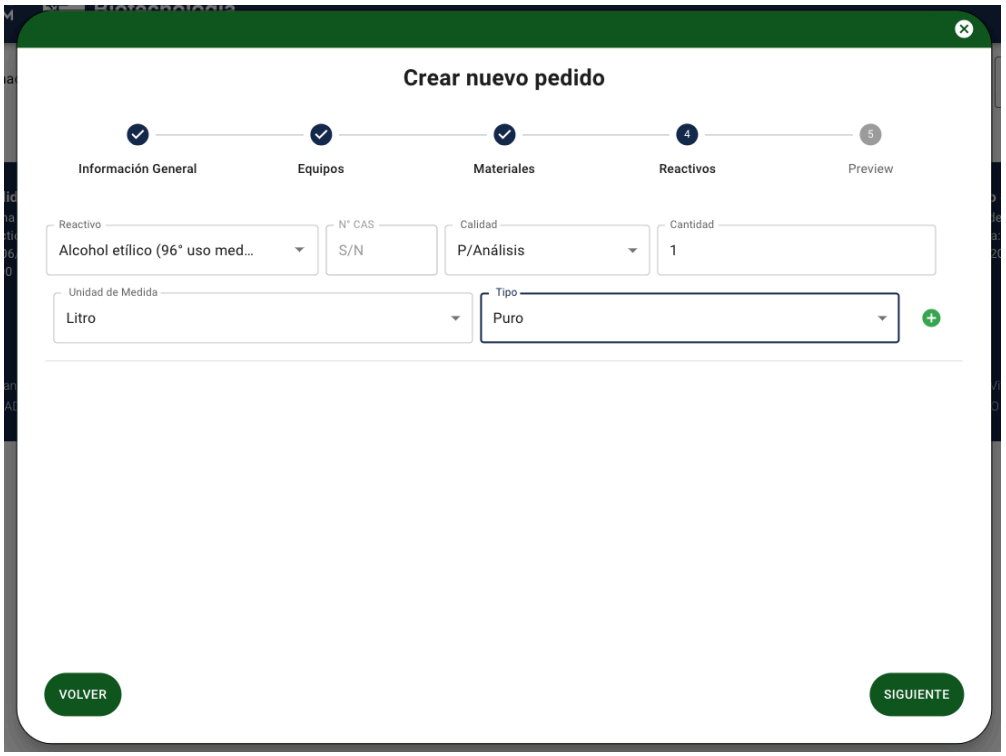


Vista del paso Materiales: agregación de materiales.

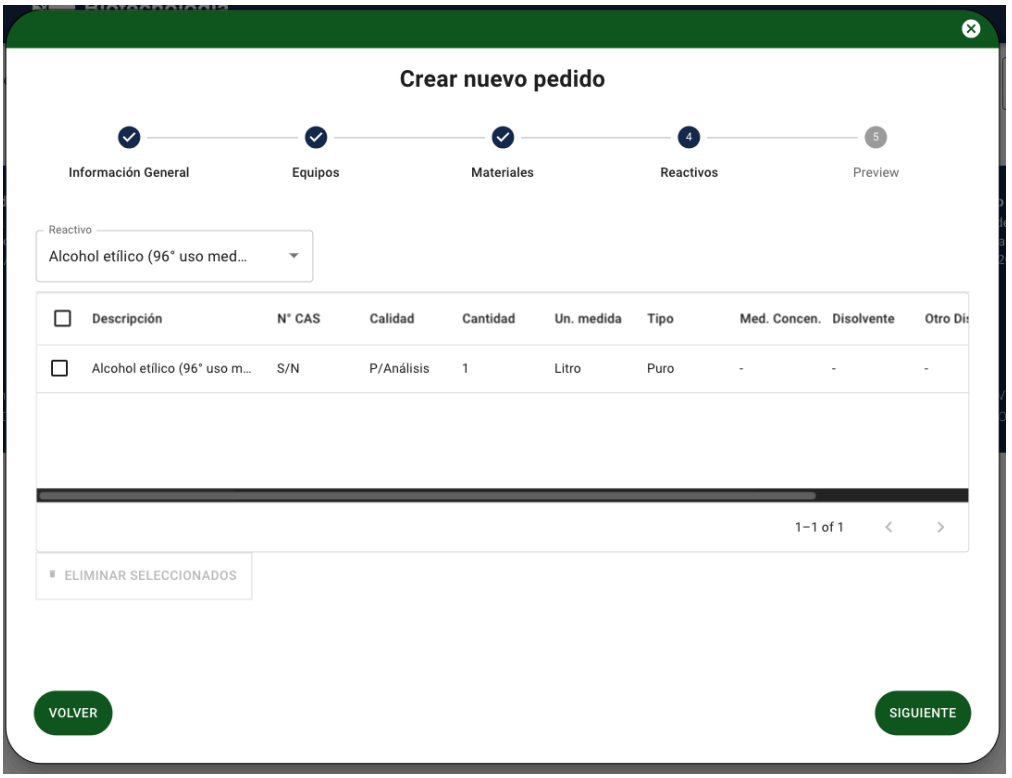


Vista del paso Materiales: con materiales agregados.

Volver

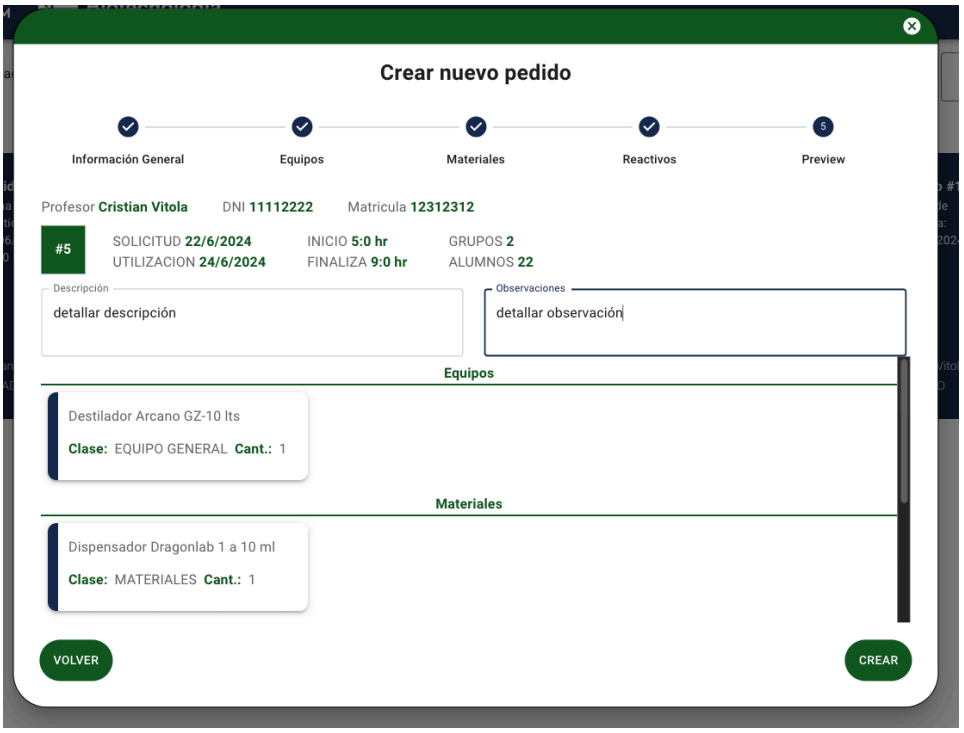


Vista del paso Reactivos: agregación de reactivos.



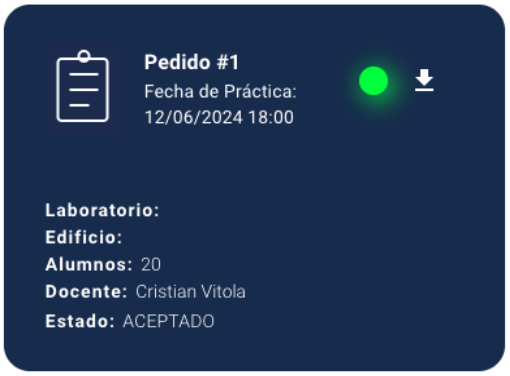
Vista del paso Reactivos: con reactivos agregados.

Volver



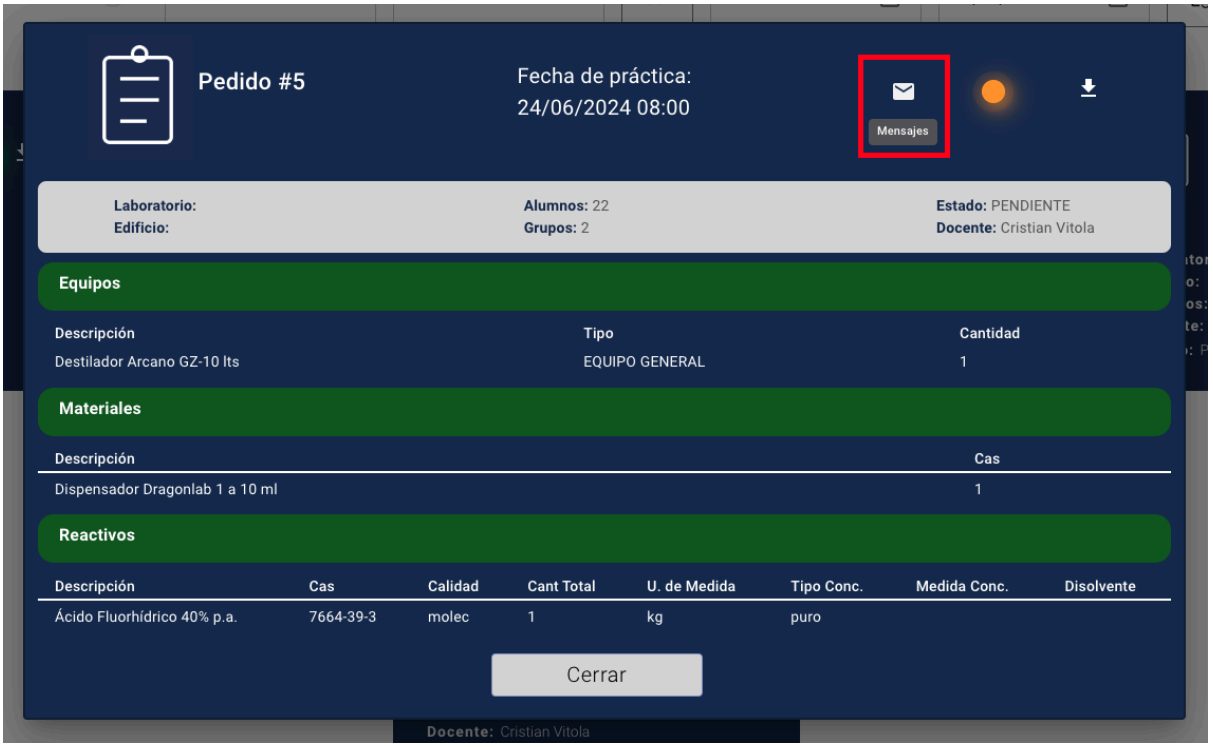
Vista del paso Preview con detalles del pedido general a ser creado.

Luego de completar toda la carga de pedidos, se presiona aceptar, para generar un pedido.

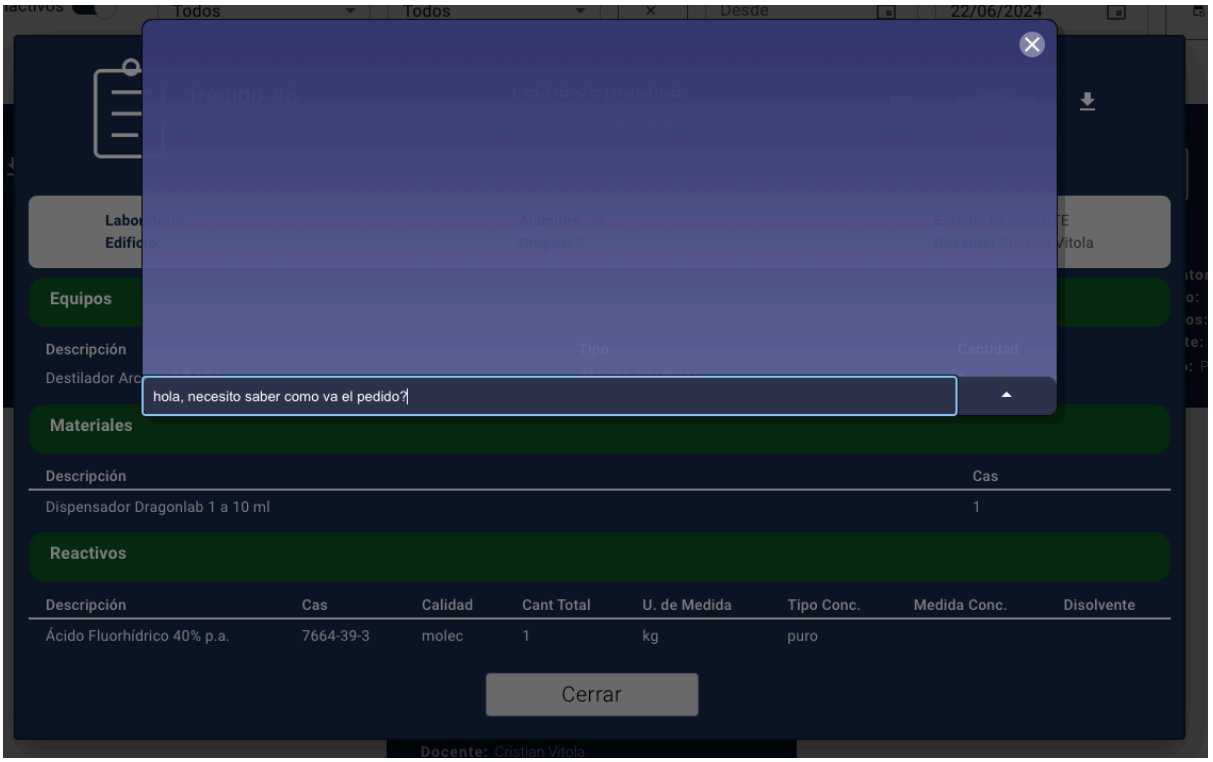


Vista del Pedido creado.

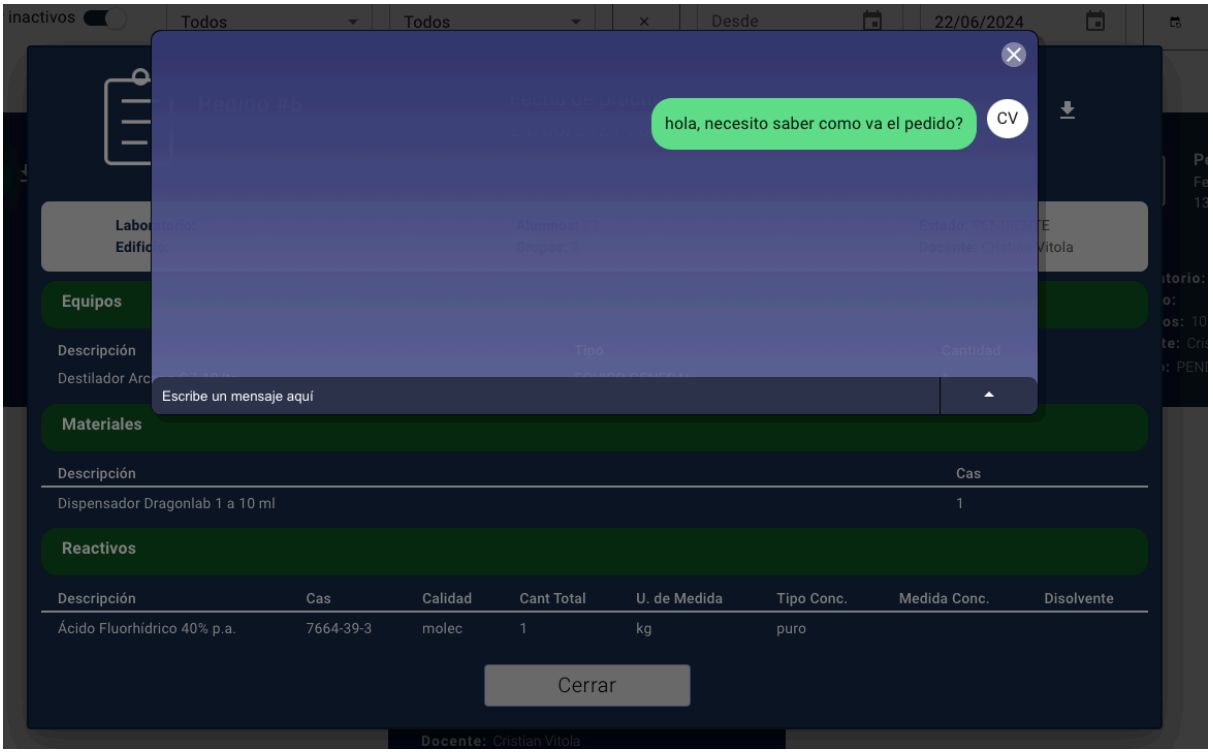
Enviar un mensaje chat	
Descripción	Permitir al usuario enviar un mensaje por medio del chat.
Condiciones	El usuario debe abrir el chat.
Entrada	Texto del mensaje nuevo
Resultado esperado	El mensaje sea enviado.
Evaluación de la prueba	Enviar mensaje correctamente.



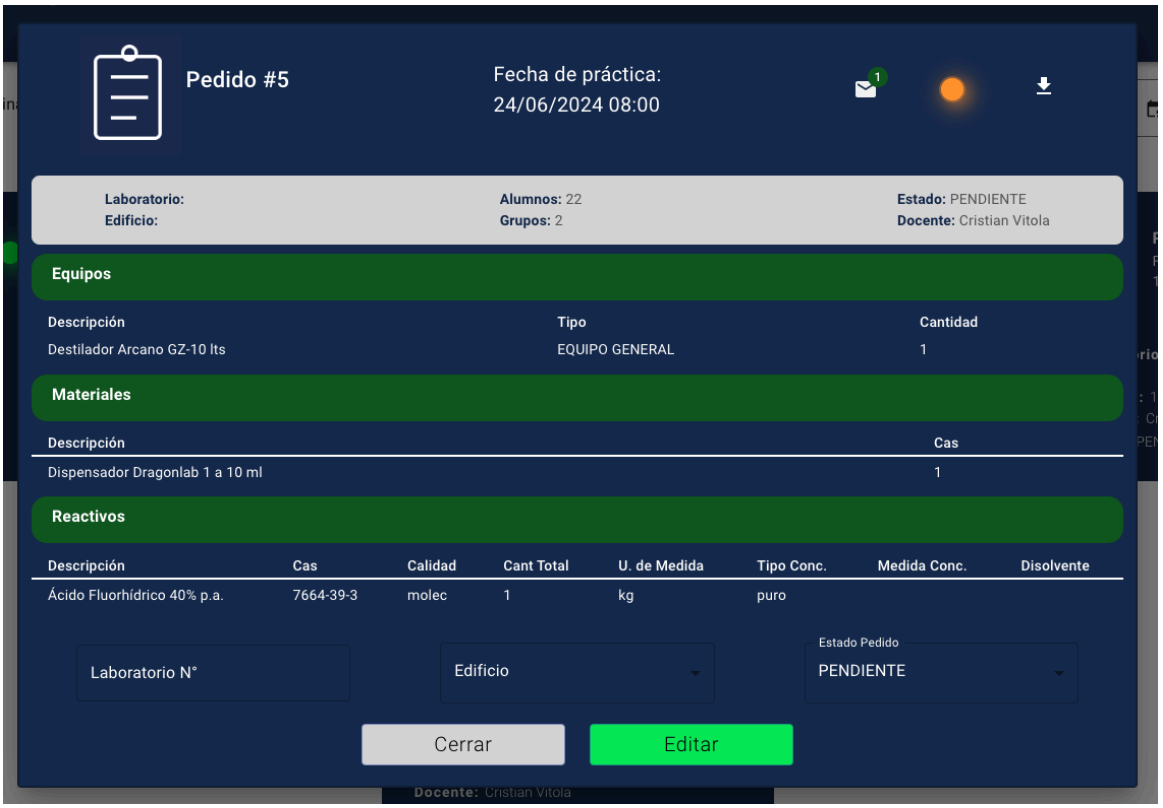
Volver



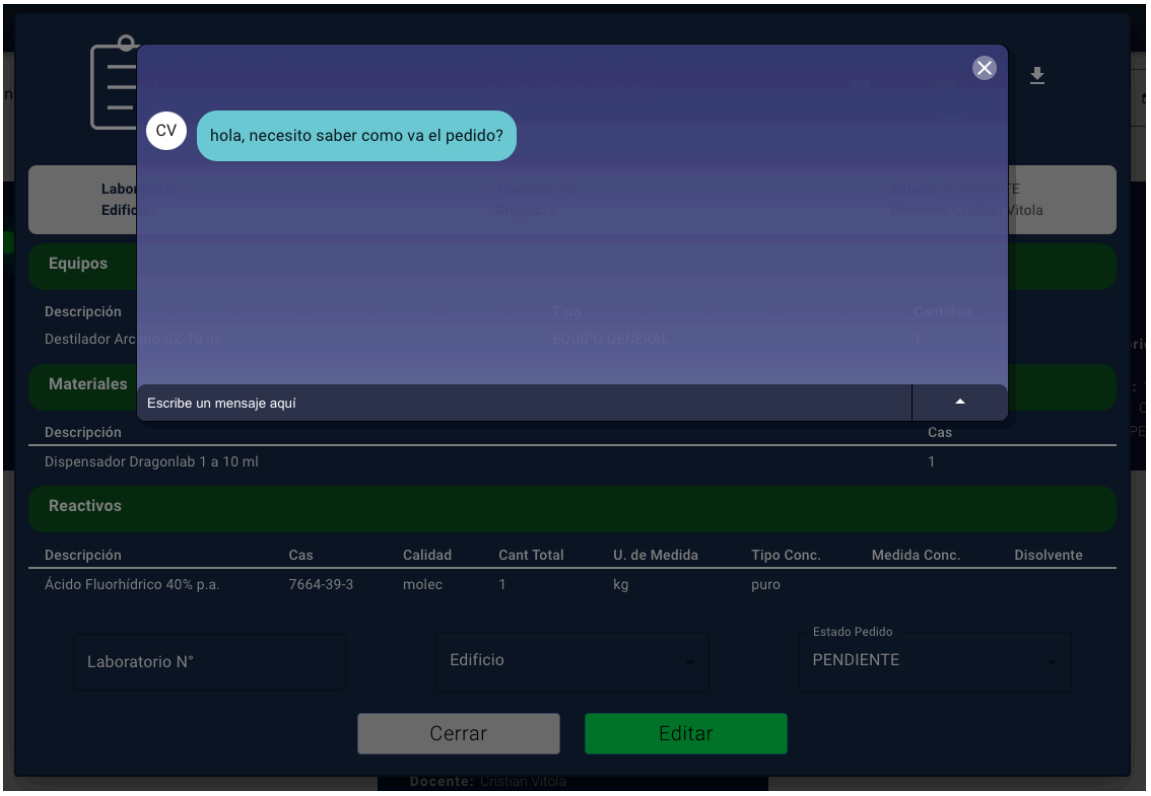
Vista de la ventana de chat de un pedido.



Vista de la ventana de chat con nuevo mensaje enviado por un perfil docente.



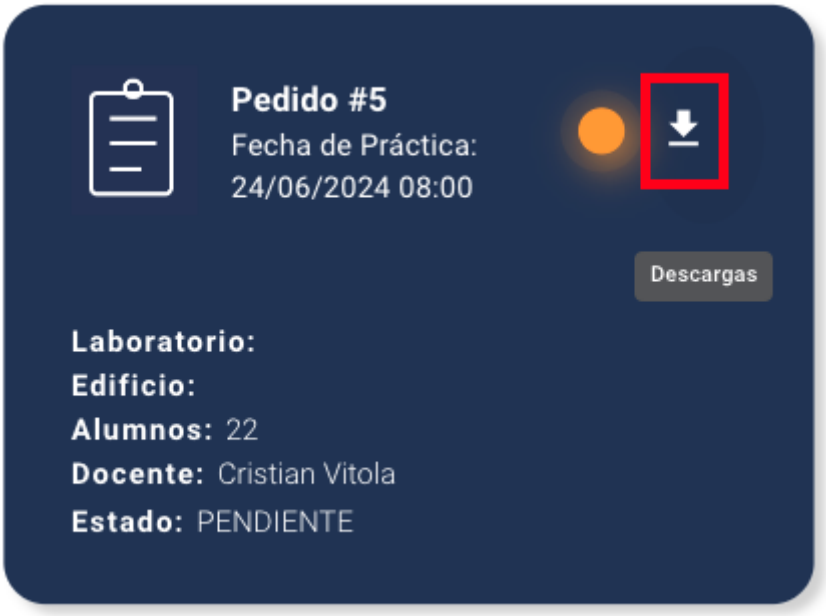
Vista de un pedido, con icono que muestra la recepción de nuevo mensaje en la vista de perfil laboratorio.



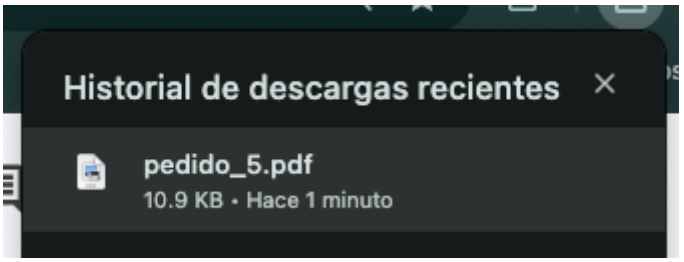
Vista de chat del perfil laboratorio, de recepción con éxito de un mensaje del perfil docente.

Volver

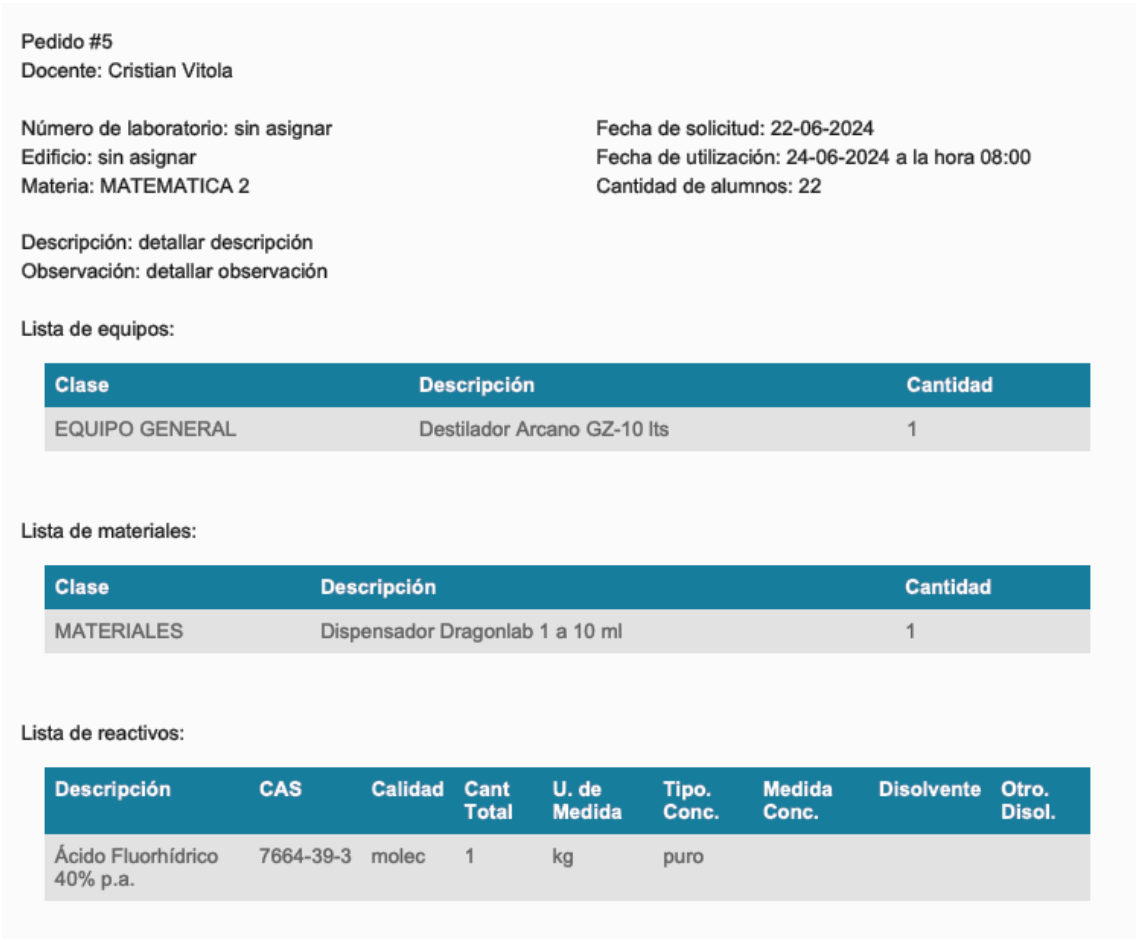
Descarga de pedido	
Descripción	Permitir al usuario poder descargar el detalle del pedido en formato PDF.
Condiciones	El usuario debe tener un pedido.
Entrada	Clave del pedido
Resultado esperado	El PDF se tiene que descargar.
Evaluación de la prueba	Descarga de PDF exitoso.



Vista de la tarjeta que se va a descargar



Vista de la descarga del archivo PDF en el navegador.



Pedido #5
Docente: Cristian Vitola

Número de laboratorio: sin asignar
Edificio: sin asignar
Materia: MATEMATICA 2

Fecha de solicitud: 22-06-2024
Fecha de utilización: 24-06-2024 a la hora 08:00
Cantidad de alumnos: 22

Descripción: detallar descripción
Observación: detallar observación

Lista de equipos:

Clase	Descripción	Cantidad
EQUIPO GENERAL	Destilador Arcano GZ-10 lts	1

Lista de materiales:

Clase	Descripción	Cantidad
MATERIALES	Dispensador Dragonlab 1 a 10 ml	1

Lista de reactivos:

Descripción	CAS	Calidad	Cant Total	U. de Medida	Tipo. Conc.	Medida Conc.	Disolvente	Otro. Disol.
Ácido Fluorhídrico 40% p.a.	7664-39-3	molec	1	kg	puro			

Vista del archivo PDF descargado.

Posibles mejoras

El Laboratorio nos había demostrado interés en que implementemos una función de mensajería por correo electrónico que envíe notificaciones tanto al docente como al Laboratorio al realizar un pedido y un resumen diario en caso de tener mensajes nuevos sin leer. Esta función debe enviar un email en la ocasión que se realice un pedido o si hay mensajes en la cola del chat. En caso de que sea una cola de mensajes del chat se enviará el email en un horario determinado, por única vez dando aviso de que tiene mensajes sin leer.

Por nuestra parte, consideramos que se podría mejorar los requerimientos mínimos solicitados a la hora de definir una contraseña de usuario. Por ejemplo, que cuente con un mínimo de 8 carácteres, al menos una mayúscula, un número y un caracter especial.

También se podría desarrollar un control de auditorías, para contar con un historial de las acciones que realizan los usuarios en la aplicación.

Volver

Sería útil tener un sistema de estadísticas, para que el Laboratorio pueda saber cuáles son los recursos más solicitados por los docentes y poder gestionar de manera más eficiente el presupuesto y el tiempo de la Universidad.

Documento de despliegue

Para poder inicializar el proyecto se van a necesitar las siguientes herramientas:

GitHub: Es una herramienta para el control de versiones distribuidas, en el cual los desarrolladores pueden hacer seguimiento del código fuente y también permite trabajar de forma colaborativa.

Aquí se encuentra información introductoria sobre github: Github

NodeJS: La instalación de Node.js es fundamental para poder publicar, compartir e instalar paquetes de software, ya sea en el registro público de npm o en registros privados. Esto te permite aprovechar el ecosistema de JavaScript y acelerar el desarrollo de tus aplicaciones.

Link de descarga de: Node.Js

Visual Studio Code: Es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. Si bien esta instalación es opcional, sugerimos que se cuente con esta herramienta para facilitar el trabajo y el manejo del proyecto.

Link de descarga de: VSCode

MongoDB Compass: Es una herramienta interactiva gratuita para consultar, optimizar y analizar datos en MongoDB, el mismo es opcional.

Link de descarga de: MongoDBCompass

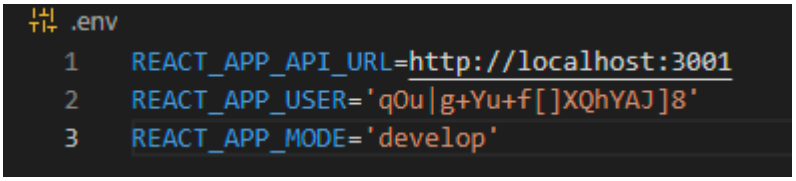
Repositorio Front-end

Para descargar y levantar el proyecto en localhost se deben realizar las siguientes instrucciones:

1. Abrir una terminal de comandos.
2. Clonar el repositorio en la terminal con el comando git clone
   Link de repositorio: Frontend
3. Una vez clonado el proyecto, se debe ir a la carpeta donde se descargo y abrirlo.
4. Abrir una terminal dentro del proyecto.
5. Ingresar el comando “**`npm i --legacy-peer-deps`**” (este mismo va a instalar las dependencias necesarias para que el proyecto funcione correctamente).

Volver

6. Luego debemos crear el archivo **“.env”** , en la raíz del proyecto. Aquí vamos a tener las variables de entorno, contiene credenciales y contraseñas que se van a manejar en el la aplicación.
7. Dentro vamos a completar los siguientes datos:



- **REACT_APP_API_URL**: aquí colocar el host en el que se va a desplegar nuestro backend.
- **REACT_APP_USER**: aquí colocar una contraseña, que sirve para encriptar información de los usuarios y no pueda ser accedidos desde el localstorage del navegador. Se puede colocar cualquier valor, pero recomendamos encarecidamente que se use el sugerido en la imagen o una contraseña robusta.
- **REACT_APP_MODE**: esta clave, se utiliza en modo **’develop’**para poder crear el Admin, de la aplicación, luego se deberá deployar como **“REACT_APP_MODE=’production’**.

Repositorio Back-end

Para descargar y levantar el proyecto en el localhost se debe realizar las siguientes instrucciones:

1. Abrir una terminal de comandos.
2. Clonar el repositorio en la terminal con el comando git clone.
   Link de repositorio: Backend

3. Una vez clonado el proyecto, se debe ir a la carpeta donde se descargo y abrirlo.
4. Abrir una terminal dentro del proyecto.
5. Ingresar el comando **“npm i --legacy-peer-deps**” (este mismo va a instalar las dependencias necesarias para que el proyecto funcione correctamente).
6. Luego debemos crear el archivo **“.env”** , en la raíz del proyecto. Aquí vamos a tener las variables de entorno, contiene credenciales y contraseñas que se van a manejar en el la aplicación.

Volver

7. Dentro vamos a completar los siguientes datos:

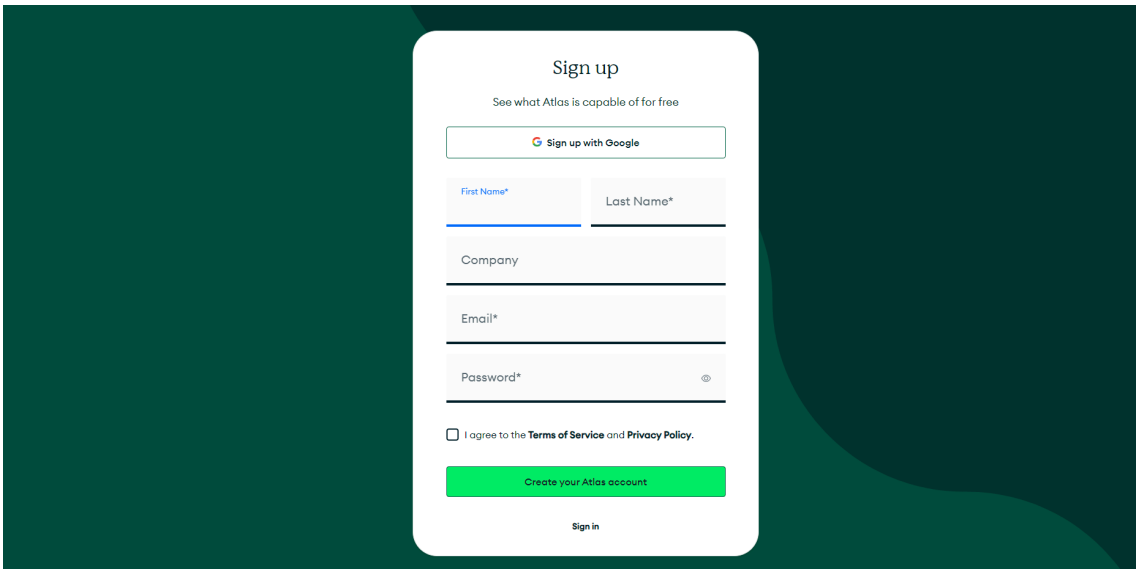
```
.env
URI_MONGO = 'url de Base de Datos'
PORT = 3001
ORIGIN1 = http://localhost:3001
ORIGIN2 = http://localhost:3000
JWT_SECRET = 'dK|}GUyjp*B*jrpS'
```

- **URI_MONGO**: aquí colocar el link donde está alojada la base de datos de la aplicación.
- **PORT**: el puerto en el cual va a correr el backend.
- **ORIGIN1**: representa URL para la lista blanca de control. Estas van a ser URL permitidas para interactuar con el backend. Es una medida de seguridad para impedir que desde otro punto de acceso pueda hacer consultas a la BD.
- **ORIGIN2**: representa URL para la lista blanca de control. Estas van a ser URL permitidas para interactuar con el backend. Es una medida de seguridad para impedir que desde otro punto de acceso pueda hacer consultas a la BD.
- **JWT_SECRET**: Aquí se coloca la clave secreta, que encripta todas las peticiones del frontend hacia el backend. Se puede colocar cualquier valor, pero recomendamos encarecidamente que se use el sugerido en la imagen o una contraseña robusta.

Creacion de Base de Datos en Mongo DB

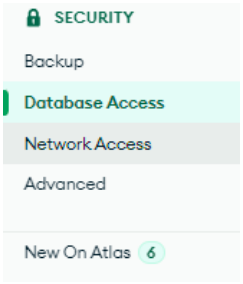
Para poder utilizar el proyecto debemos crear una Base de datos no relacional en MongoDB

1. Dirigirnos a la página de MongoDB e ingresamos al entorno con nuestras credenciales, si no se cuenta con las mismas, se deberá crear una nueva cuenta.
Link de MongoDB.

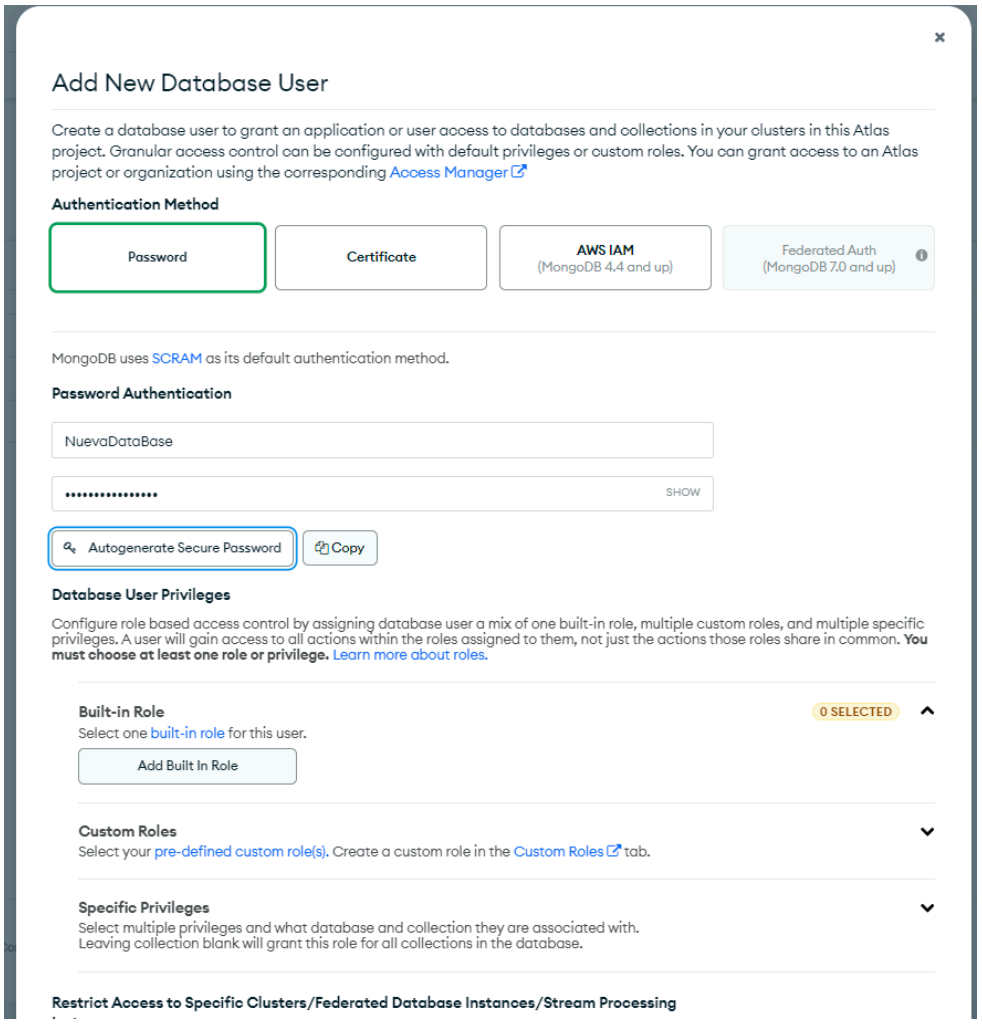


Volver

2. Hacemos click en Database Access en la sección SECURITY

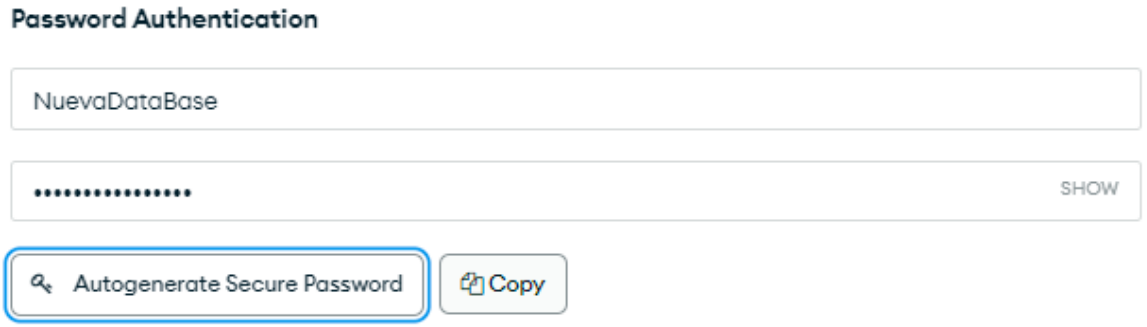


3. Al costado derecho nos va a aparecer un botón de +ADD NEW DATABASE USER. Este nos abre la siguiente ventana.

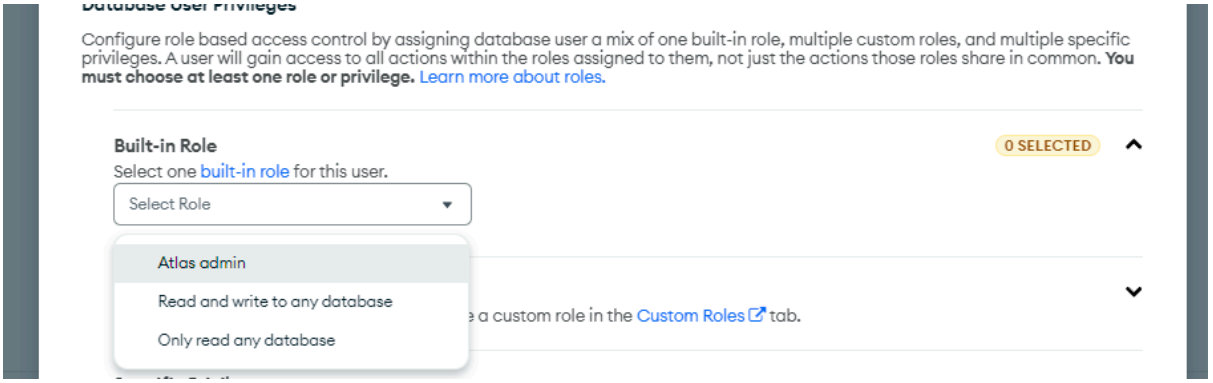


Volver

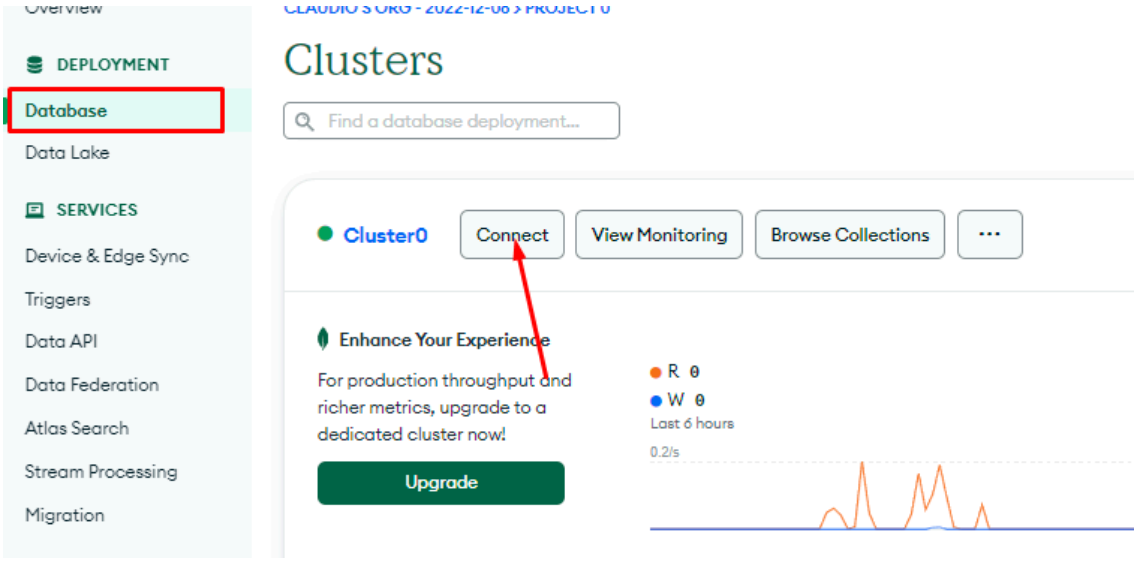
4. Completamos los siguientes campos, con el nombre, que representa al usuario con acceso qué va administrar la base de datos, y la clave, que la auto generamos con el botón debajo (al hacer click en **SHOW**, podremos visualizarla). Estas credenciales, debemos guardarlas para usar más tarde.



5. Lo siguiente es asignar **Built-In-Role**, y luego aceptamos estos cambios con el botón **Add User**.

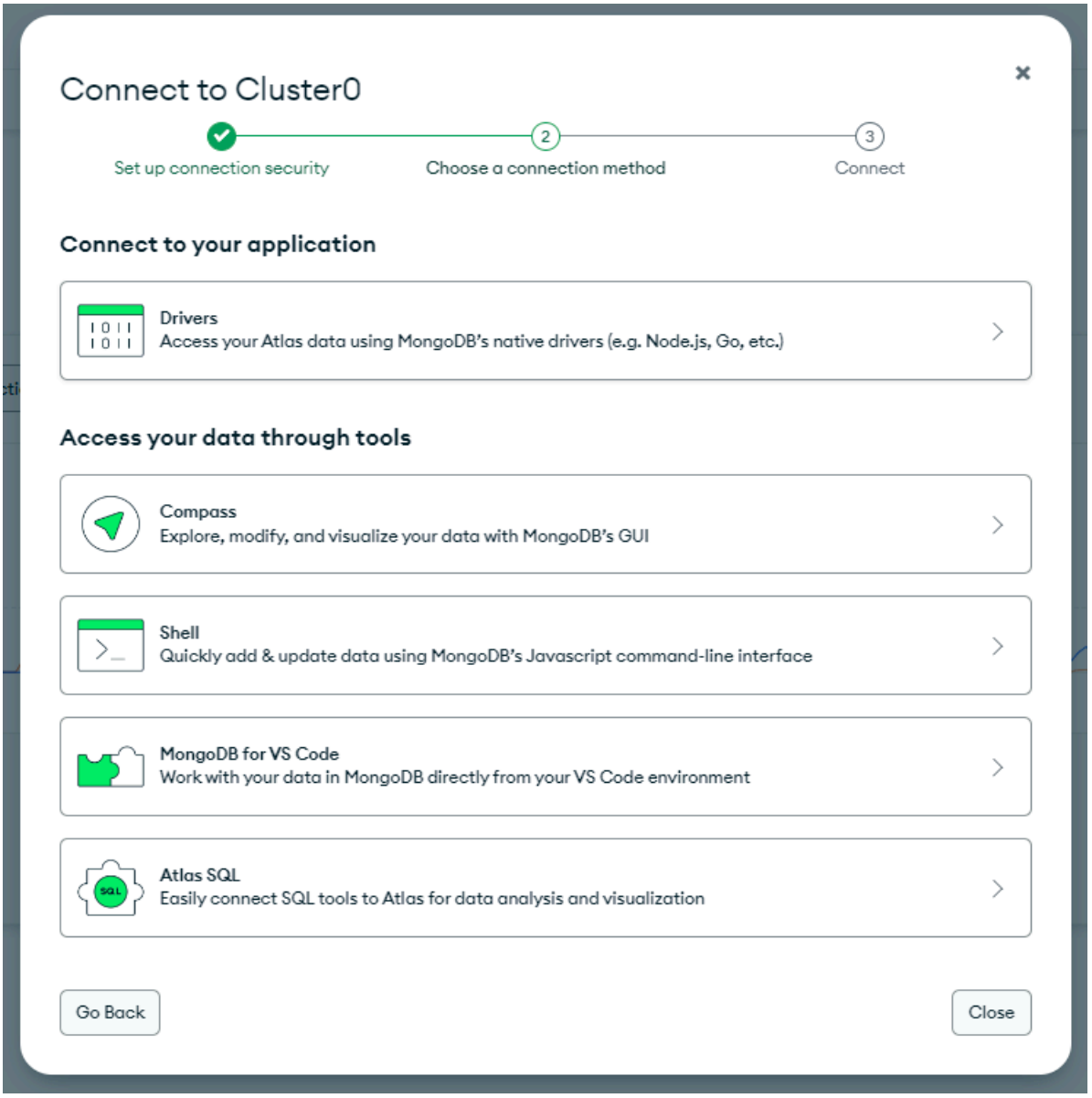


2. Dirigirnos a la sección Database, y luego hacemos click en Connect.



Volver

3. Hacemos click en la sección Connect to your application - Drivers:

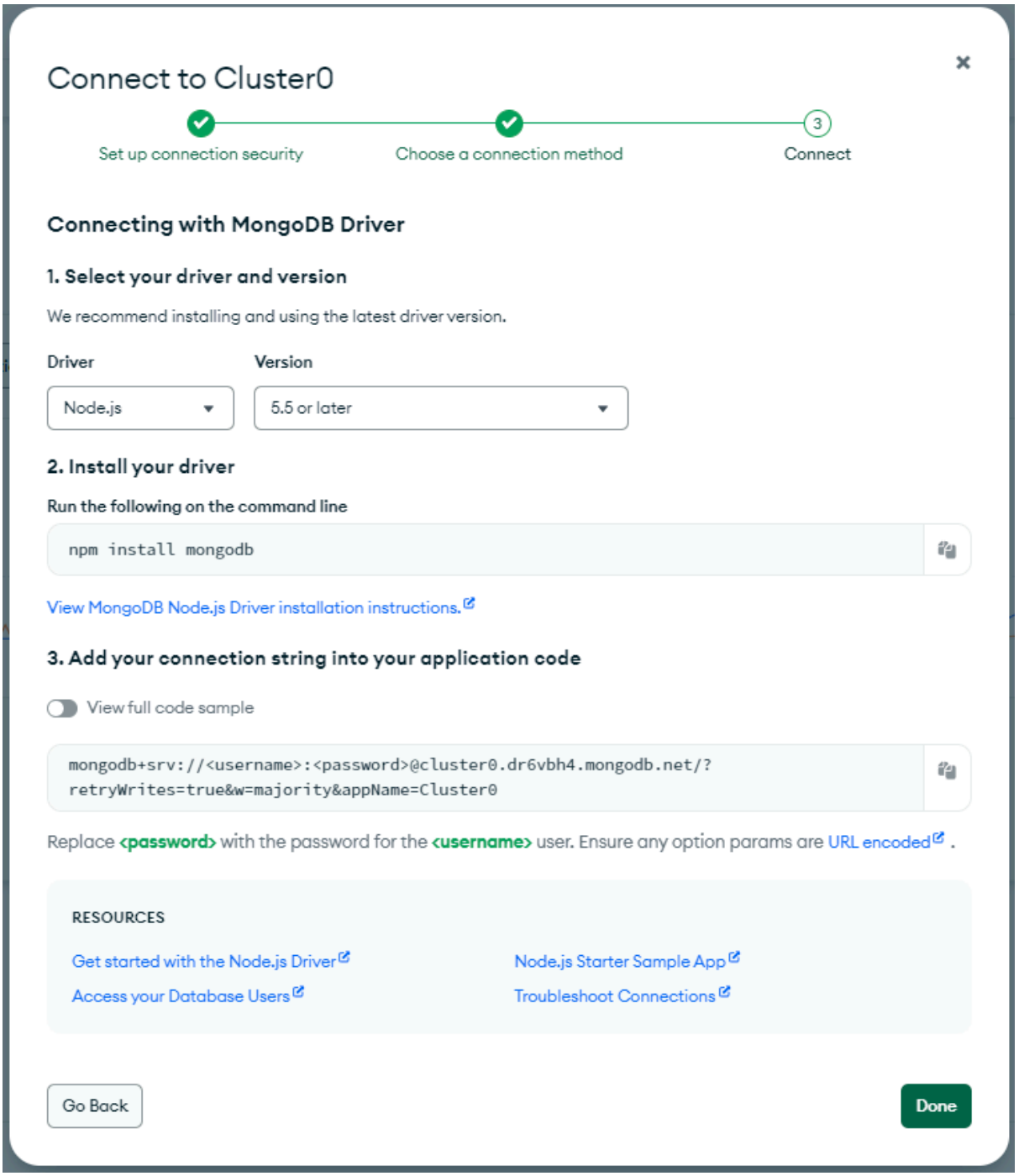


Volver

4. En la ventana siguiente nos va aparecer la url de la base de datos, a la cual luego tenemos que modificar con las credenciales que generamos antes, pero solo vamos a necesitar la primera parte de la url.

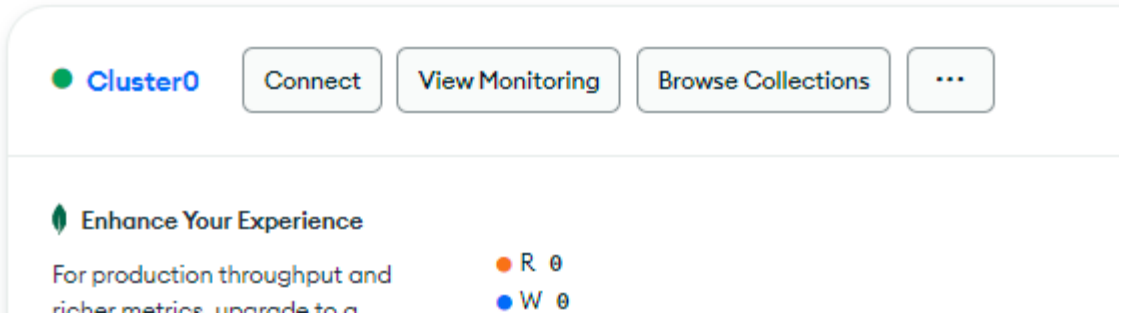
`mongodb+srv://<username>:<password>@cluster0.dr6vbh4.mongodb.net/`

Lo copiamos y reemplazamos <username> y <password>, por el nombre y password generados anteriormente y presionamos **Done**.

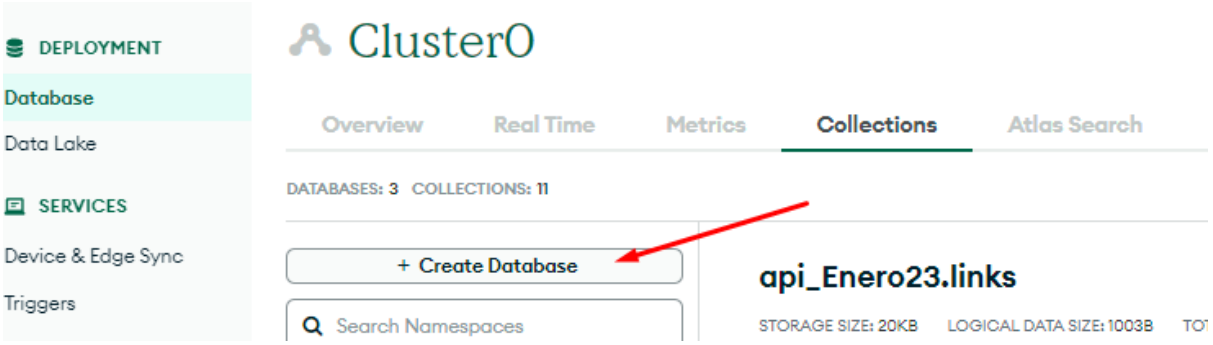


Volver

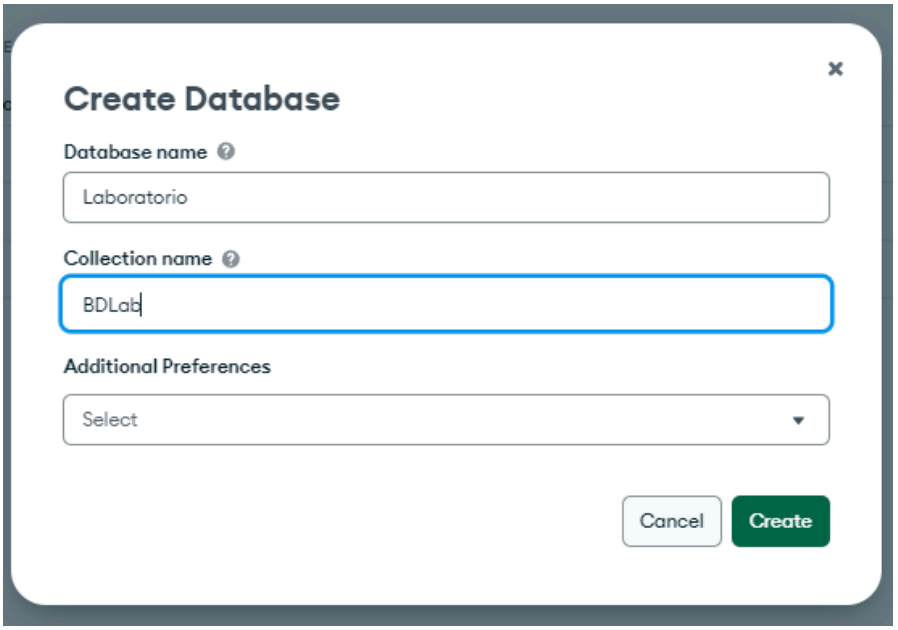
6. Lo siguiente es presionar en **Cluster0**, es un espacio gratuito, que nos permite hacer hasta 5000 peticiones por mes. Si se requiere aún más, debe contratar un servicio Cluster de pago(+ CREATE).



7. Presionamos + Create Database



8. Se nos pide completar 2 campos, el primero es el nombre de la BD, y el segundo es el nombre que va a tener nuestra colección de datos dentro de la BD. Por último damos Create.



Volver

9. Volvemos al archivo **.env** del Backend y completamos la clave **URI_MONGO**, con la URL de mongoDB que armamos con el nombre y password.

mongodb+srv://<username>:<password>@cluster0.dr6vbh4.mongodb.net/

Iniciar el proyecto localmente

Para poder levantar el proyecto, solo resta abrir dos terminales, navegar y entrar a las carpetas, del frontend y backend descargadas en cada una de ellas, y correr el comando **npm start**

Creación del perfil Admin

Una vez creada la base de datos, y levantado el proyecto tanto el backend como el frontend, mongoose creará la estructura lógica para operar y solo pueden crearse usuarios a través de un Admin, entonces, debemos crear uno:

1. Dirigirse al archivo .env del Front-end y poner **REACT_APP_MODE='develop'** guardar y correr **npm start** para levantar el proyecto.
2. Entrar a la web de MongoDB. y dirigirse a la colección creada previamente.
3. Agregar en la lista de "usuarios", la siguiente estructura, se debe respetar los tipos en color gris, presionar en INSERT DOCUMENT:
   - _id: **<dejar el que esta por defecto>**,
   - usuario: "<nombre de usuario del Admin>" (String)
   - contrasenia: **"$2b$10$TrsBSx5JPJx8U9gQazlrF.4JMKEQ6dglANaWlUU3U8Wq2mkP6xdG6"** (String)
   - nombre: "<nombre>" (String)
   - apellido: "<apellido>" (String)
   - dni: "<dni>" (String)
   - matricula: "<matricula>" (String)
   - admin: **true** (Boolean)
   - email: "<e-mail>" (String)
   - editor: false (Boolean)
   - __v: 0 (Boolean)
   - rol: **"lab"** (String)

     Todos los marcados con **negrita**, deben estar como se indica.
4. Pulsar Insert.
5. Listo, ahora ya se puede logear en el sistema con el usuario que se creó y utilizar la contraseña **123123**
6. Se procede a cambiar la contraseña por una a elección, Para esto dirigirnos a la sección **"USUARIOS"**, en el sistema.
7. Hacer click en el lápiz del perfil Admin creado, para editarlo.
8. Ingresamos nueva contraseña y Aceptar.

Volver

9. Por último, cambiamos el estado del archivo **.env** del Front-End por el siguiente **REACT_APP_MODE='production'**

Desplegar proyecto en Internet (ejemplo Render)

FRONT-END

1. Iniciar sesión en Render
2. Apretar en +NEW y se pulsar la opción de Web Service
3. Elegimos la opción Build and deploy from a Git repository y presionamos Next.
4. En Connect Repository, buscamos la ruta del repositorio del frontend.
5. Configurar el despliegue en la sección Settings.
    - Build Command: npm install && npm run build
    - Start Command: npm run start
6. Configurar el Environment, acá debemos crear las variables de entorno que teníamos en el archivo .env:
    - **REACT_APP_API_URL**: aquí colocar el host en el que se va a desplegar nuestro Backend, como todavía no lo tenemos, por ahora lo dejamos vacío. Una vez desplegado el Backend, copiamos la url y la copiamos en esta clave.
    - **REACT_APP_USER**: contraseña de encriptación de usuarios.
    - **REACT_APP_MODE**: Si el admin, aun no se creo en la Base de datos de Mongo, seguir los pasos de Creación del perfil Admin.
7. Presionar al botón Create Web Service.
8. Cuando termine el proceso, nos va a asignar una URL para nuestro proyecto, que vamos a usar para acceder a él, Esta Url debe ser una de las 2 claves ORIGIN del backend.

BACK-END

1. Volvemos a la pantalla de inicio.
2. Apretar en +NEW y se pulsar la opción de Web Service
3. Elegimos la opción Build and deploy from a Git repository y presionamos Next.
4. En Connect Repository, buscamos la ruta del repositorio del frontend.
5. Configurar el despliegue en la sección Settings.
    - Build Command: npm install && npm run build
    - Start Command: npm run start
6. Configurar el Environment, acá debemos crear las variables de entorno que teníamos en el archivo .env:
    - **URI_MONGO**: aquí colocar el link donde está alojada la base de datos de la aplicación.
    - **PORT**: el puerto en el cual va a correr el backend.
    - **ORIGIN1**: URL del frontend.
    - **ORIGIN2**: URL del backend, por ahora la dejamos vacía, hasta que Render, nos genere una al finalizar.
    - **JWT_SECRET**: Aquí se coloca la clave secreta, que encripta todas las peticiones del frontend hacia el backend.

Volver

7. Por último, debemos dar al botón Create Web Service.
8. Al finalizar Render nos genera una URL para el backend, que ya podremos usar para reemplazar el la clave **`REACT_APP_API_URL`** del frontend.

Volver

Conclusión final

A lo largo de la cursada de esta materia, hemos enfrentado numerosos desafíos que nos han permitido crecer tanto profesional como personalmente. Desde el principio, al tratarse de un proyecto heredado con un stack tecnológico inicial ya definido, tuvimos la oportunidad de conocer y trabajar con nuevas tecnologías.

Para la mayoría de nosotros, fue nuestro primer acercamiento a tecnologías como Material UI y MongoDB. Esta experiencia nos ha permitido expandir nuestro conocimiento y habilidades en áreas que antes no habíamos explorado.

La dinámica de trabajo grupal nos presentó problemáticas interesantes, especialmente para aquellos de nosotros que no habíamos trabajado previamente en la industria. Aprender a trabajar en conjunto, organizar las tareas, coordinar los desarrollos y asignar responsabilidades fue un proceso enriquecedor. La necesidad de adaptarnos y colaborar de manera efectiva nos ayudó a desarrollar competencias esenciales para el trabajo en equipo y la gestión de proyectos.

Además, la experiencia de trabajar de manera "freelancer" nos planteó un nuevo conjunto de desafíos. Tuvimos que acercar nuestro desarrollo al laboratorio, presentar nuestras propuestas y avances, y al mismo tiempo, recibir su feedback y conocer sus necesidades. Este intercambio constante nos permitió formular nuevos requerimientos para el sistema y trabajar sobre ellos, para poder asegurar que el desarrollo se alineará con las expectativas y demandas del laboratorio.

La cursada también nos enriqueció en aspectos más personales, especialmente en lo que respecta a la realización de presentaciones frente a profesores y compañeros. Exponer el trabajo realizado no fue tarea fácil, especialmente si consideramos que en nuestra área no estamos acostumbrados a hablar frente a un grupo de personas, y menos de manera presencial. Sin embargo, esta experiencia nos ayudó a mejorar nuestras habilidades de comunicación y a ganar confianza en nuestra capacidad para presentar y defender nuestro trabajo.

A mitad de camino, incorporamos un nuevo integrante al equipo, lo que desencadenó un nuevo proceso de integración. Realizamos el onboarding del nuevo compañero, introduciéndolo tanto en las generalidades (objetivo del proyecto, la metodología de trabajo del grupo y el funcionamiento general del aplicativo) como en las cuestiones específicas (organización del código, el funcionamiento de los endpoints y el despliegue de la aplicación). Este proceso no solo nos permitió tener un mayor entendimiento del dominio del problema, sino que también nos orilló a compartir y explicar nuestro desarrollo a alguien que no conocía el contexto específico. La incorporación de Agustín fue un impulso positivo para el equipo, y su entusiasmo y habilidades complementaron perfectamente nuestras dinámicas de trabajo.

En conclusión, este proyecto no solo nos ha permitido desarrollar una aplicación exitosa, sino que también ha consolidado nuestro equipo y nuestras capacidades para

Volver

enfrentar futuros desafíos. Nos vamos con una profunda satisfacción y el orgullo de haber culminado estos diez sprints con éxito. La experiencia de trabajar con nuevas tecnologías, resolver problemas en equipo, y adaptar nuestras propuestas a las necesidades del laboratorio ha sido invaluable. Nos llevamos de este proyecto una serie de aprendizajes que sin duda nos fortalecerán en nuestro camino profesional.

Volver