ACÁMICA - DWFS
Bloque 3

# DELILAH RESTÓ

**Sofía Varela**

# DEFINIR PRIORIDADES

## PASO 0

Repaso general de toda la teoría del bloque.
Repaso git.

## PASO 1



Revisar Guías + Checklist y **definir prioridades** -> **QUÉ NECESITO**

## PASO 2

Armar un plan de acción -> **CÓMO LO VOY A LOGRAR**

## PASO 3

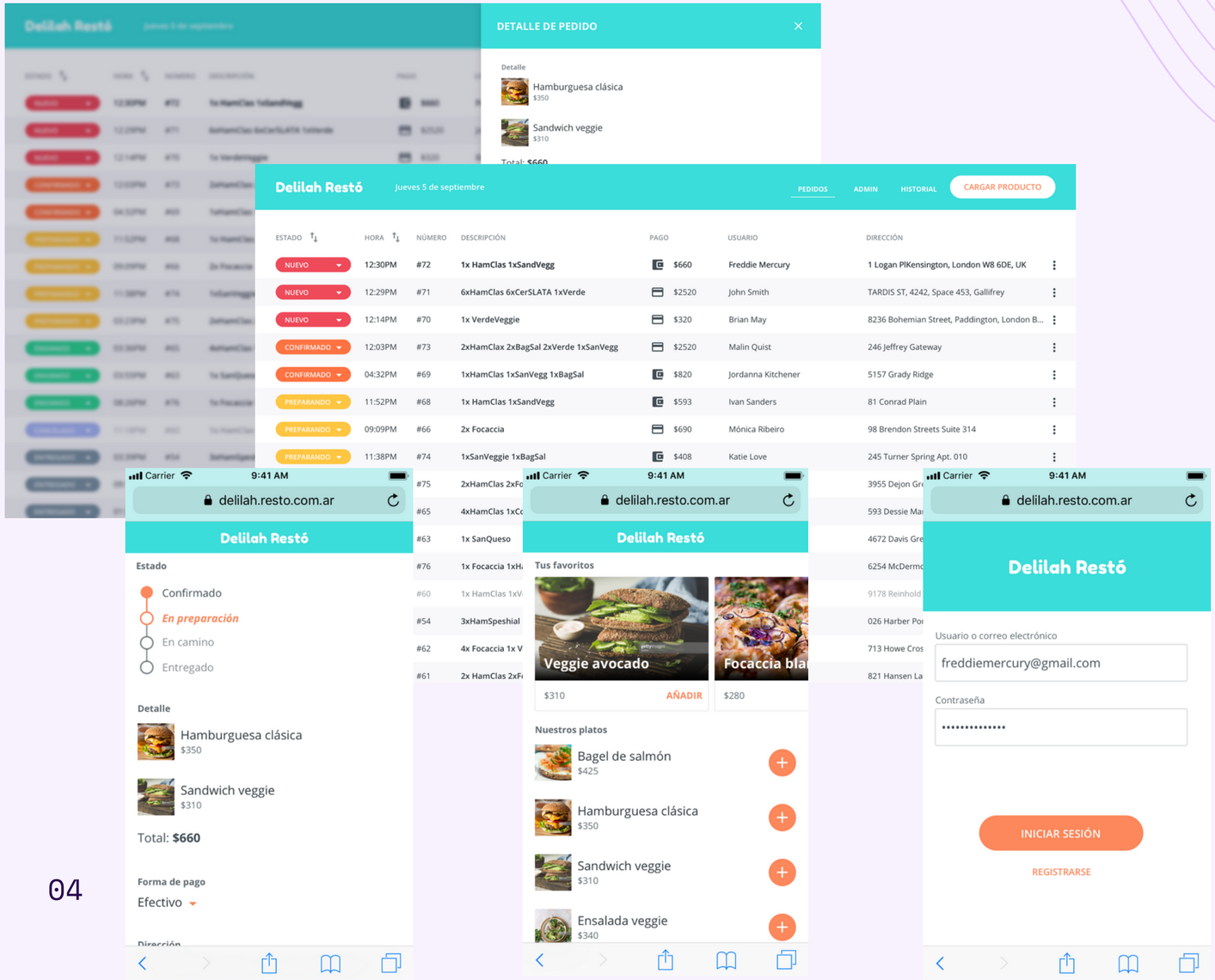
Ejecutarlo! Con objetivos claros y bien definidos.

# ANALIZANDO LAS VISTAS

**OBJETIVOS:**
-**P**lanificar al máximo mi DB y las relaciones funcionales entre las tablas
-> Definir Diagrama entidad relación DER
-Especificar los endpoints requeridos

EVITAR REINGENIERÍA DURANTE EL PROCESO

# ANALIZANDO LAS VISTAS: DB




## TABLA PRODUCTOS

Productos disponibles en el restó. Disponibles y no disponibles

## TABLA USUARIOS

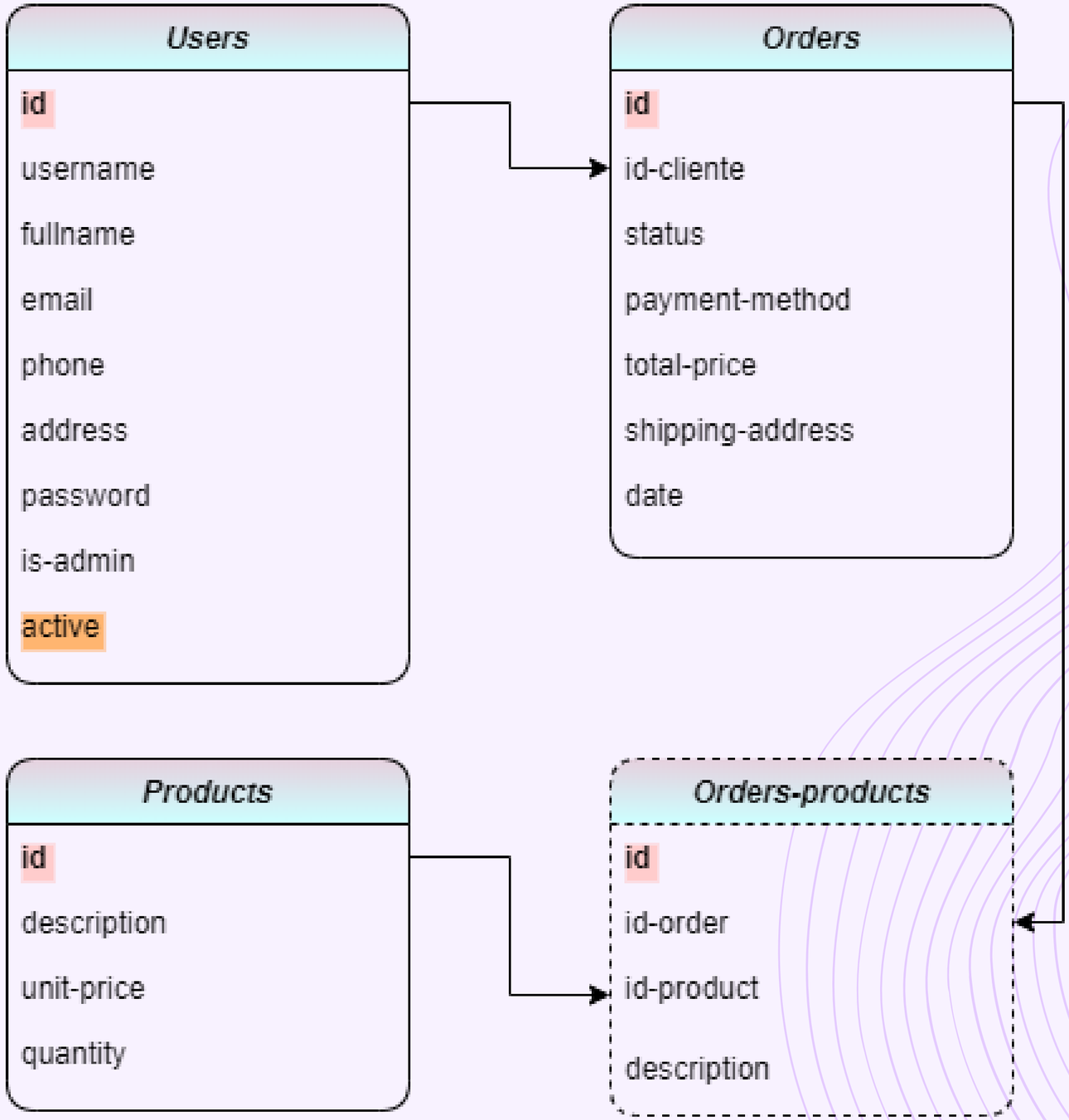
Usuarios dados de alta en el sistema. Incluye un administrador que creo por default, y usuarios regulares que pueden darse de alta. Dificultad: validación y autenticación. Autorización según tipo usuario.

## TABLA PEDIDOS

Con fecha, descripción, usuario que lo hizo, métodos de pago.

## TABLA PEDIDOS-PRODUCTOS

JOIN de las tablas de pedidos y productos. Con una descripción que incluye todos los productos del pedido.




## TABLA PRODUCTOS

**Productos disponibles en el restó. Disponibles y no disponibles**

## TABLA USUARIOS

**Usuarios dados de alta en el sistema. Incluye un administrador que creo por default, y usuarios regulares que pueden darse de alta. Dificultad: validación y autenticación. Autorización según tipo usuario.**

## TABLA PEDIDOS

**Con fecha, descripción, usuario que lo hizo, métodos de pago.**

## TABLA PEDIDOS-PRODUCTOS

**JOIN de las tablas de pedidos y productos. Con una descripción que incluye todos los productos del pedido.**

# CREACIÓN DE TABLAS

```sql
DROP TABLE IF EXISTS products;
DROP TABLE IF EXISTS users;
DROP TABLE IF EXISTS orders;
DROP TABLE IF EXISTS orders_products;

CREATE TABLE products(

    id INT UNSIGNED NOT NULL auto_increment,
    description VARCHAR(255) NOT NULL,
    unit_price INT NOT NULL,
    quantity INT UNSIGNED NOT NULL,
    available BOOLEAN DEFAULT TRUE,

    PRIMARY KEY (id)
);
```

```sql
CREATE TABLE users(

    id INT UNSIGNED NOT NULL auto_increment,
    username VARCHAR(255) NOT NULL,
    fullname VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    phone INT UNSIGNED NOT NULL,
    address VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL,
    is_admin BOOLEAN DEFAULT FALSE,
    active BOOLEAN DEFAULT TRUE,

    PRIMARY KEY (id)
);
```

```sql
CREATE TABLE orders(

    id INT UNSIGNED NOT NULL auto_increment,
    id_cliente INT UNSIGNED,
    status ENUM('NUEVO', 'CONFIRMADO', 'PREPARANDO',
    'ENVIANDO', 'CANCELADO', 'ENTREGADO'),
    payment_method ENUM('credit-card','cash',
    'debit-card'),
    total_price INT NOT NULL,
    shipping_address VARCHAR(255) NOT NULL,
    date_order DATE,

    PRIMARY KEY (id)
);
```

```sql
CREATE TABLE orders_products(

    id INT UNSIGNED NOT NULL auto_increment,
    id_order INT UNSIGNED NOT NULL,
    id_product INT UNSIGNED NOT NULL,
    description VARCHAR(255) NOT NULL,

    PRIMARY KEY (id)
);
```

# PLANIFICACIÓN DE ENDPOINTS - CRUD

## /PRODUCTOS

**GET:** obtener listado completo;
**GET by ID o nombre:** obtener producto en particular;
**POST:** *admin*-agregar producto;
**DELETE**: *admin*-eliminar producto;
**PATCH**: admin-agotar producto o volver a estar disponible;
**PUT**: *admin*-modificar información del producto.

## /USUARIOS

**POST**: crear nuevo usuario;
**GET**: *admin*-obtener listado completo de usuarios. Ordenar alfabéticamente.
**GET**: *no admin*-obtener datos propios como usuario, solo si están logueados;
**PUT**: editar datos de usuario;
**PATCH**: *admin*-hacer administrador a otro usuario o deshacerlo;
**DELETE**: desactivar un usuario, darlo de baja.

## /USUARIOS/LOGIN
## /USUARIOS/LOGOUT
## /USUARIOS/REGISTER

## /PEDIDOS

**POST:** crear nuevo pedido;
**GET:** *admin*-obtener listado completo de pedidos de todos los usuarios. Ordenar por hora o status (/pedidos?sort_by=date);
**GET**: no *admin*-obtener listado de todos los pedidos propios;
**PATCH:** *admin*-cambiar el status;
**PUT**: hacer un cambio en el pedido (*solo* si no está ya en preparación), ej agregar producto;
**DELETE**: cancelar el pedido (solo si no está ya en preparación).

## /PEDIDOS/HISTORIAL

**Seleecionar fecha y poder listar todos por horario. Admin solo.**

# Iniciando el proyecto

- NPM INIT -Y

Inicio proyecto. Creación package.json

- DEPENDENCIAS NECESARIAS

Express, body-parser, jsonwebtoken, sequelize, mysql2

- PLAN/MAPEO DE PROYECTO

.js para la app/servidor + .sql con la creación de las tablas + .js de modelos de las entidades que tengo. Separación del proyecto en carpetas: a analizar.

# PRÓXIMOS PASOS: LARGO CAMINO POR DELANTE

## -DOCUMENTACIÓN API

Swagger: escribir la especificación en YAML

## -ESTABLECER CONEXIÓN CON SERVIDOR Y LA BASE DE DATOS

## -CODEAR ENDPOINTS Y MIDDLEWARES SEGÚN DOCUMENTACIÓN API

Middlewares para inicio sesión, para borrar usuario, para pedir producto ver que no esté agotado, para agotar producto ver que no esté previamente agotado, para crear usuario ver que no se repita, endpoint comidas no mostrar una que no este disponible, etc.
Un middleware general para toda la app.

## -JSON WEB TOKEN

Usar token para inicio sesión: autenticación y autorización según tipo usuario: admin o no.

# PRÓXIMOS PASOS: LARGO CAMINO POR DELANTE

## -TESTEAR Y VERIFICAR CONEXIONES A SERVIDOR Y DB Y CORRECTO FUNCIONAMIENTO

Postman. Probar alta usuarios, listar comidas por pedido, listar productos por orden según status y según fecha, etc.
Que solo admin pueda ver la vista de todos los pedidos, que solo admin pueda agregar alimentos. solo usuarios administradores puedan crear, editar y eliminar productos, y que los usuarios logueados solo tengan acceso a su información personal.

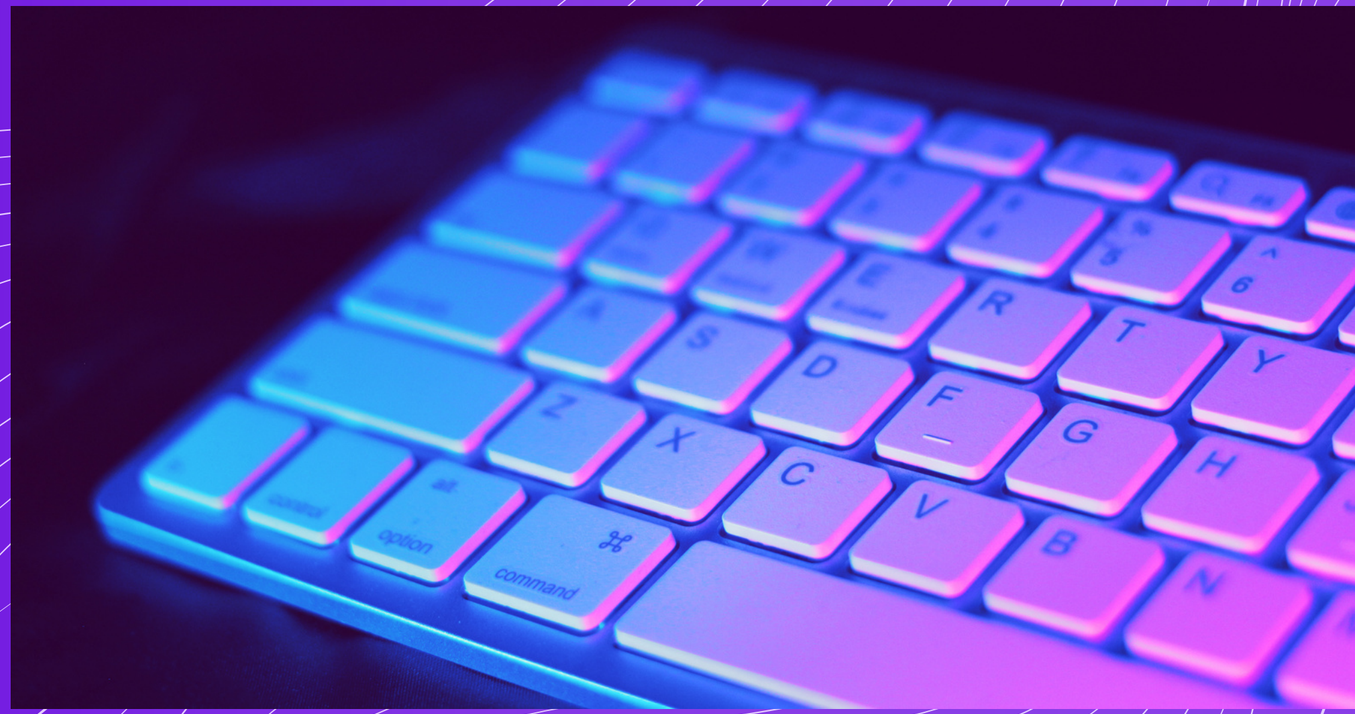
## -DEFINIR

-.gitignore tener en cuenta para node_modules;
-usar replacements por seguridad;
-ver cómo consolidar Descripción en Orders_products;
-Definir criterio para "TUS FAVORITOS";
-definir endpoint para el admin ver los pedidos ordenados según status o según fecha:

```
SELECT *
FROM Orders
ORDER BY date DESC;
```

# MUCHAS GRACIAS!

NO HAY PRREGUNTAS?
NO HAY PREGUNTAS :)




(SÍ HAY ESPECIAL AGRADECIMIENTO A TOD@S USTEDES POR EL SOPORTE Y AYUDA CONTINUOS)