Universidad de Los Andes

Facultad de Ingeniería

Escuela de Sistemas

Departamento de Computación

ALL IN ONE: PROYECTO FINAL DE BASES DE DATOS.

Desarrollado por:

Barrios Yurley

C.I. 22.986.327

Mérida, Julio de 2019

PROPUESTA DEL PRODUCTO

All In One es una aplicación web que permite a usuarios registrados, guardar, buscar y compartir recetas de cocina. De manera que puedan tener un Cookbook personalizado y con sus mejores recetas de forma ordenada. El usuario tendrá la posibilidad no solo de ver las recetas agregadas por él mismo, si no también por otro usuarios y poder guardarlas en sus favoritos para así consultarlas directamente desde su perfil cada vez que lo desee.

OBJETIVOS

- Almacenar información de recetas añadidas por usuarios de diversas partes del mundo.
- Diseñar un sitio donde las recetas puedan ser almacenadas y localizadas de manera rápida y organizada.

ALCANCE DEL PRODUCTO

El alcance de la aplicación tendrá los siguientes aspectos:

- El usuario podrá registrarse a través de un username y contraseña.
- El usuario podrá iniciar sesión con su información de registro.
- El usuario tendrá la opción de agregar sus propias recetas a su perfil.
- El usuario podrá ver, editar y eliminar sus recetas agregadas.
- El usuario podrá buscar recetas agregadas por otros usuarios
- El usuario podrá guardar en sus favoritos las recetas que más le interesen.

DEFINICIÓN DE LA ARQUITECTURA

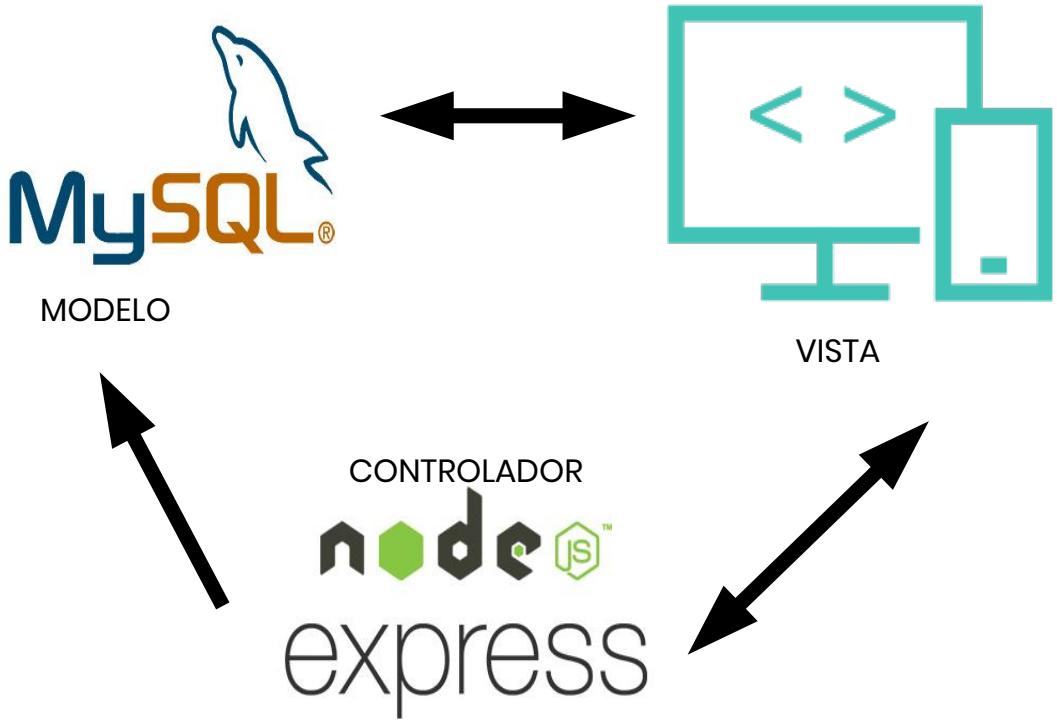

Figura 1. Patrón Arquitectónico

Arquitectura. La arquitectura de la aplicación consta de varias capas, las cuales representan los componentes físicos, un servidor y varios clientes.

- Capa de presentación: es la visual que tiene el usuario de la aplicación, con un mínimo de desarrollo, a través de una interfaz web en HTML, aquí se captura la información, se comunica con las otras capas y muestra el resultado obtenido.

- Capa de lógica: escrita en código JavaScript a través de NodeJS contiene las operaciones de recepción y devolución de datos y procesamiento de información, se comunica con la capa de datos para solicitar o actualizar información y con la capa de presentación para enviar los eventos a mostrar. En esta capa se valida la información capturada en la capa de presentación.

- Capa de datos: es donde residen los datos de la aplicación, en la que se utiliza MySQL para el manejo de persistencia de datos, recibe las solicitudes de almacenamiento y actualización de los datos provenientes de las otras capas.

Patrón Arquitectónico. Los patrones arquitectónicos son estrategias de diseño empleadas para aprovechar las buenas practicas reconocidas en procesos y experiencias que han sido efectivas y reutilizables, el diseño web de la aplicación All in One utiliza el patrón arquitectónico de Modelo-Vista-Controlador, el cual permite realizar programación multicapa.

Así pues la vista es la página en sí donde el usuario puede interctuar con los diferentes componentes. El modelo es el sistema de

gestión de base de datos MySQL y el controlador encargado de procesar las interacciones del usuario y ejecuta los cambios adecuados en el modelo o en la vista, a través de NodeJS, quien se encarga de realizar la peticiones y dar respuestas.

DEFINICIÓN DE LENGUAJES Y ENTORNO DE DESARROLLO

El principal lenguaje de programación implementado es JavaScript, el cual es un lenguaje de programación interpretado, orientado a objetos,basado en prototipos, imperativo y dinámico. Este lenguaje posee un entorno un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor, el cual es Node JS, este entorno asíncrono, con I/O de datos en una arquitectura orientada a eventos y será el que permita conectar la aplicación con la base de datos diseñada.

En cuanto a la base de datos, se utilizará el modelo relacional, implementando como sistema de gestión a MySQL, el cual está desarrollado en su mayor parte en ANSI C y C++.

Por su parte, el entorno de desarrollo es el conjunto de herramientas necesarias para programar aplicaciones, a continuación se describe los componentes a utilizar en el proyecto All in One:

-Sublime Text: Editor de código fuente en C++ y Python para los plugins. Gracias a estos plugins se puede escribir código de una manera más ágil y ordenada.

-Consola / Terminal: Será utilizada para correr el código y conocer los mensajes y errores de parte del servidos.

-NPM: Es el sistema de gestión de paquetes por defecto para Node.js, un entorno de ejecución para JavaScript y permite instalar y ejecutar los diferentes componentes, frameworks y módulos a necesitar.

-Express: Express es un framework web transigente, escrito en JavaScript y alojado dentro del entorno de ejecución NodeJS. Proporciona mecanismos para:

- Escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL (rutas).
- Integración con motores de renderización de "vistas" para generar respuestas mediante la introducción de datos en plantillas.
- Establecer ajustes de aplicaciones web como qué puerto usar para conectar, y la localización de las plantillas que se utilizan para renderizar la respuesta.
- Añadir procesamiento de peticiones "middleware" adicional en cualquier punto dentro de la tubería de manejo de la petición.

-Módulos Utilizados: Gracias a NodeJS y Express se pueden utilizar módulos que permitan realizar de manera mucho más ágil los procesos, estos son:

- express-handlebars: Permite utilizar plantillas de HTML para hacer más eficaz la creación de las vistas y la interfaz de usuario.

- express-session mysql: Permite almacenar las diferentes sesiones de usuarios en la base de datos.

- express-mysql-session passport: Middleware de autenticación para Node, que junto con Sequelize y MySQL permite implementar el registro y el inicio de sesión del usuario.

- Passport-local: Middleware de autenticación que permite gestionar la autenticación de los usuarios.

- timeago.js: Permite convertir el formato de tiempo recibido por el usuario a uno mucho más amigable.

- Connect-flash: permite mostrar mensajes en la pantalla bajo ciertas condiciones, estos mensajes sirven para indicar al

usuario cuando ha realizado una actividad satisfactoriamente o cuando algo ocurrió mal.

- Express-validator: Permite validar la información del usuario.

-Dia: Utilizado para la creación de los esquemas y diagramas relacionados con el diseño de la base de datos.

ENFOQUE Y METODOLOGIA

Enfoque y método seguido

Se ha considerado las fases del proyecto según la definición PMBOCK

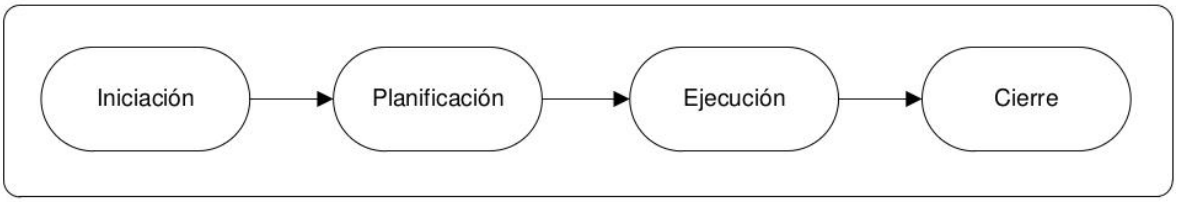

Figura 2. Enfoque PMBOCK

Dentro de cada fase, se definen las acciones siguientes:

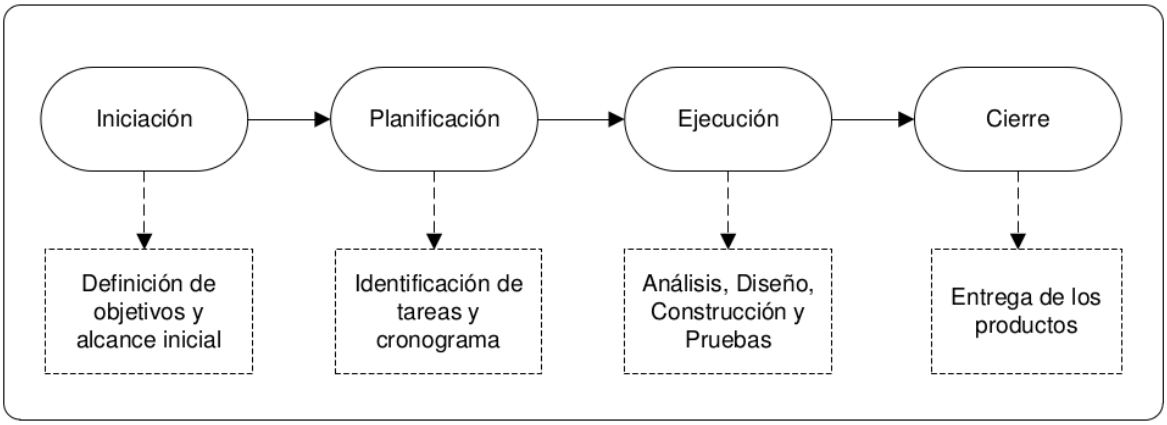

Figura 3. Fases de metodología.

A continuación se describe las diferentes fases y acciones:

Iniciación: Definición de los objetivos y el alcance inicial.

Planificación: Identificación de tareas, estimación de esfuerzo para cada tarea y cronograma. Identificación de riesgos y plan de contingencia.

Análisis: Definición del Alcance Detallado, Diagrama de Casos de Uso y Modelo Conceptual de la Base de Datos.

Diseño: Definición del Modelo Lógico de la Base de Datos e identificación de las estructurasnecesarias (procedimientos, funciones, triggers.).

Construcción: Desarrollo de las sentencias SQL que permiten la construcción de las tablas y demás estructuras de la Base de Datos.

Pruebas: Ejecución de los guiones de pruebas.

Cierre: Entrega de productos finales.

ESPECIFICACIÓN DE REQUERIMIENTOS

Los requerimientos constituyen las especificaciones en cuanto a características y funcionalidades propuestas en éste proyecto, se clasifican en requerimientos funcionales y no funcionales.

Requerimientos Funcionales. Los requerimientos funcionales están enfocados al comportamiento interno de la aplicación. En éste proyecto se tendrán los siguientes requerimientos funcionales:

- REQF-1. El sistema debe permitir el registro de los usuarios.
- REQF-2. El sistema debe permitir el inicio de sesión de usuarios registrados.
- REQF-3. El sistema debe permitir agregar una nueva receta.
- REQF-4. El sistema debe permitir ver las recetas guardadas.
- REQF-5. El sistema debe permitir editar y eliminar las recetas guardadas.
- REQF-6. El sistema debe permitir buscar recetas por titulo y/o categorías.
- REQF-7. El sistema debe permitir visualizar las recetas buscadas.

- REQF-8. El sistema debe permitir visualizar las últimas recetas agregadas.
- REQF-9. El sistema debe permitir visualizar todas las recetas que se encuentran en la plataforma.
- REQF-10. El sistema debe permitir al usuario agregar a sus favoritos recetas de otros usuarios.
- REQF-11. El sistema debe permitir al usuario cerrar sesión en su cuenta de All in One.

Requerimientos no Funcionales. Los requerimientos no funcionales están relacionados con las características generales del sistema, hacen referencia a factores como la seguridad, integridad, usabilidad, escalabilidad, etc... En este proyecto se tendrán los siguientes requerimientos no funcionales:

- REQNF-1. El sistema debe asegurarse de tener los permisos necesarios para el tratamiento de datos personales del usuario.
- REQNF-2. El sistema debe estar implementado en javaScript
- REQNF-3. El sistema debe tener un diseño de interfaz amigable e intuitivo.
- REQNF-4. El sistema debe encriptar las contraseñas de usuarios para

preservar su seguridad.

-REQNF-5: El sistema debe validar el registro e inicio de sesión de los usuarios.

Actores del Sistema. Los actores son los elementos que interactúan con la aplicación, para éste proyecto se han definido los siguientes actores:

- El usuario de la aplicación: el usuario final quien podrá utilizar todas las opciones de la aplicación.
- El sistema: la aplicación que en nombre del usuario podrá interactuar con los datos del usuario

Listado de Casos de Uso. Los casos de uso son las situaciones de interacción entre el usuario y la aplicación, otorga un sentido realista de lo que el usuario puede hacer con las opciones de la aplicación, se construyen a partir de los requerimientos dados, a continuación el listado de casos de uso para éste proyecto.

- CU-1. Registrarse en la plataforma.
- CU-2. Iniciar sesión.
- CU -3 Validar registro.
- CU-4 Validar inicio de sesión.

- CU-5. Agregar una nueva receta.

- CU-6. Ver recetas agregadas.

- CU-7. Editar recetas.

- CU-8. Eliminar recetas.

- CU-9 Consultar listado de últimas recetas.

- CU-10. Buscar recetas.

- CU-11. Guardar recetas favoritas

-CU-12. Cerrar sesión .

Matriz de Casos de Uso. La matriz de casos de uso relaciona cada uno de los casos de uso con los requerimientos contemplados.

CASO DE USO	NOMBRE	REQUERIMIENTO
CU-1	Registrarse en la plataforma	REQF-1
CU-2	Iniciar Sesión	REQF-2
CU-3	Validar registro	RQNF -4 RQNF -5
CU-4	Validar Inicio de sesión	RQNF -4
CU-5	Agregar una nueva receta	REQF-3
CU-6	Ver recetas agregadas	REQF-4 REQF-9
CU-7	Editar recetas	REQF-5
CU-8	Eliminar recetas	REQF-5
CU-9	Consultar ultimas recetas	REQF-8

CU-10	Buscar recetas	REQF-6 REQF-7
CU-11	Guardar recetas favoritas	REQF-10
CU-12	Cerrar sesión	REQF-11

Tabla 1. Casos de Uso

MODELADO DE REQUERIMIENTOS

Diagrama de Casos de Uso.

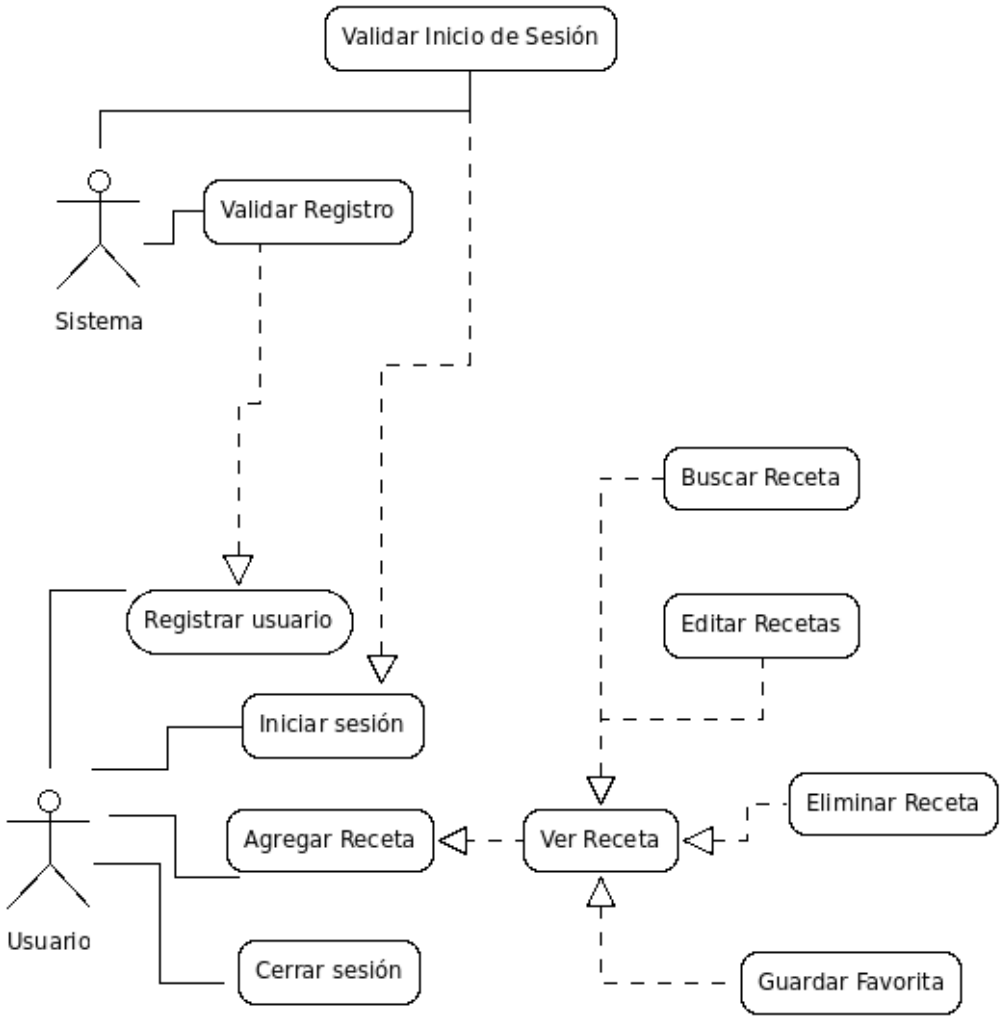

Diagrama 1. Casos de Uso

El diagrama de casos de uso es la representación gráfica de las funcionalidades de la aplicación y los actores que interactúan con ellas, en el caso de la aplicación All In One existe un usuario que es el propietario del dispositivo y accede a todas las funcionalidades, todos

los casos de uso de la aplicación son accesibles por el usuario, algunos, especificaciones de otros casos de uso y representados mediante relaciones de inclusión.

IMPLEMENTACIÓN DE REQUERIMIENTOS

Identificación de entidades y atributos

Recipes: Es la entidad principal. Es con esta entidad que los usuarios podrán interactuar para agregar la información de las diferentes recetas.

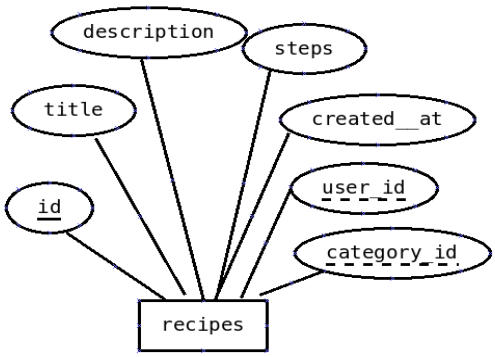

Diagrama 2. Entidad Recipes

Users: Es la entidad que permitirá la gestión de la base de datos. Son los usuarios quienes almacenarán y actualizarán la mayor parte de la información referente a las recetas.

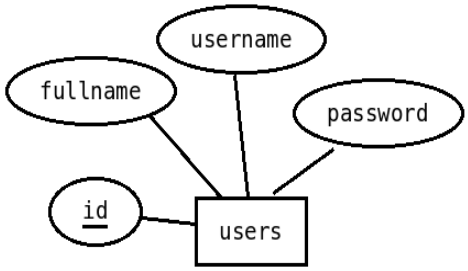

Diagrama 3. Entidad users.

Favorites: Esta entidad se relaciona con usuarios y recetas. Permitirá saber a cuáles son las recetas favoritas de cada usuario.

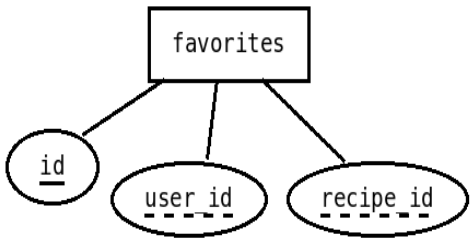

Diagrama 4. Entidad favorites

Categories: Relacionada con las recetas. Permite realizar una clasificación por categoría, de cada una de las recetas agregadas.

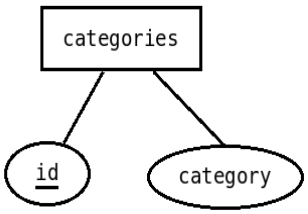

Diagrama 5. Entidad Categories

Ingredients: Relacionada con las recetas. Permitirá obtener una lista de todos los ingredientes posibles a utilizar.

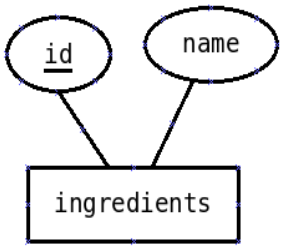

Diagrama 6. Entidad ingredients.

Measures: Los ingredientes tienen medidas y cantidades diferentes de acuerdo a cada receta.

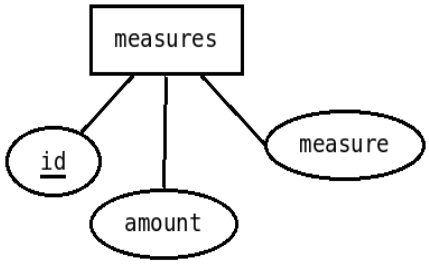

Diagrama 7. Entidad measures

Modelo Entidad Relación

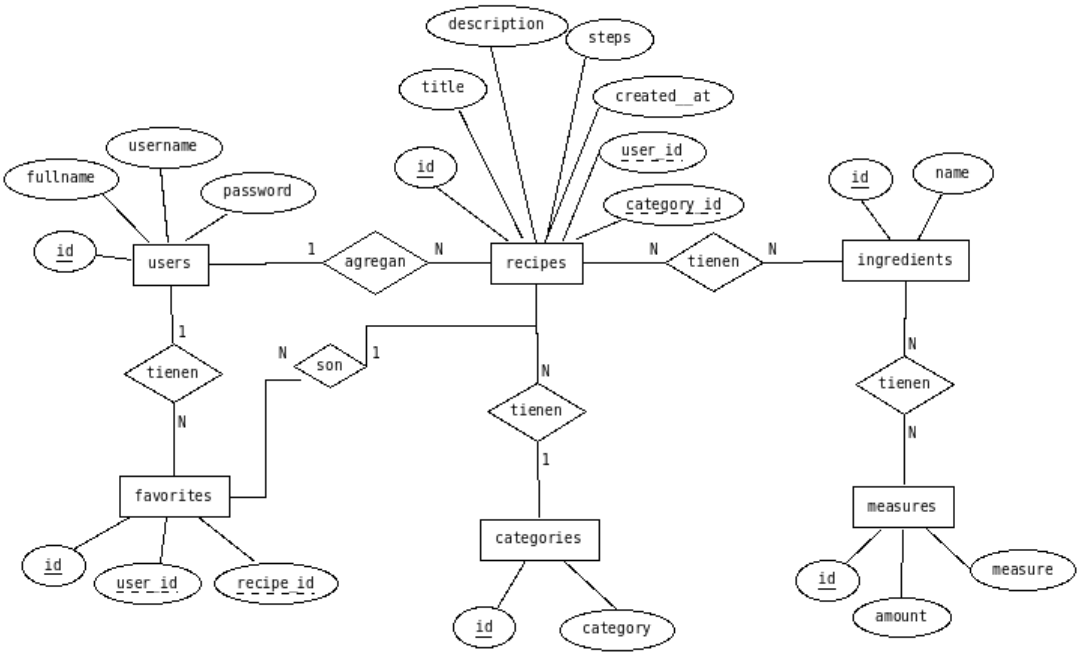

Diagrama 8. Modelo entidad relación.

El modelo entidad relación permite visualizar de manera gráfica el comportamiento de las entidades y cómo interactúan con las demás. Además de conocer la multiplicidad de cada una de las relaciones.

Modelo Físico: El modelo de datos físicos representa cómo se construirá el modelo en la base de datos.

Un modelo de base de datos física muestra todas las estructuras de tabla, incluidos el nombre de columna, el tipo de datos de columna, las restricciones de columna, la clave principal, la clave externa y las relaciones entre las tablas.

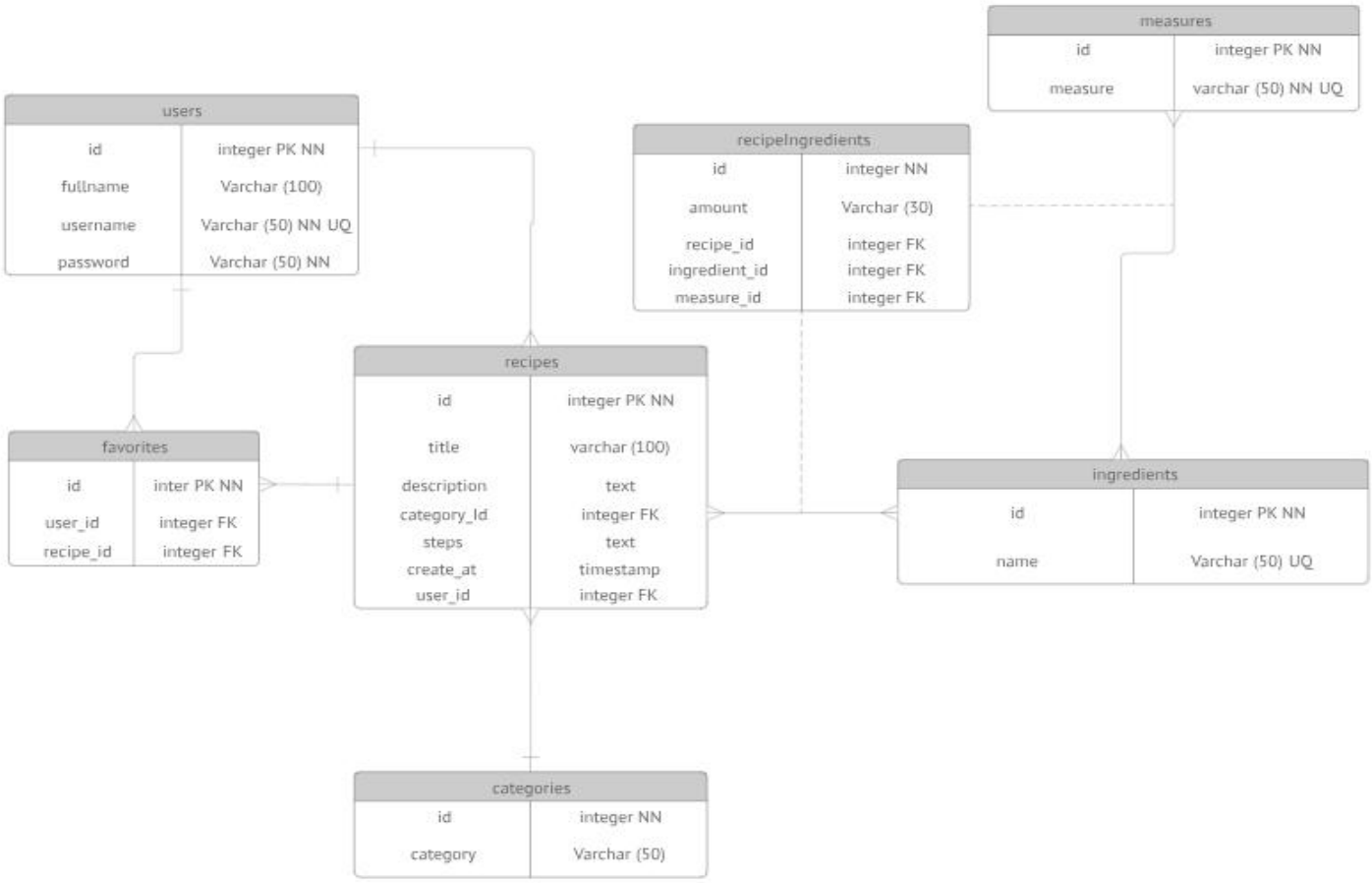

Diagrama 9. Modelo Relacional

Nota: Dado que las relaciones Recipes – Ingredients e Ingredients – Measures, son de muchos a muchos, se agregó una nueva tabla, correspondiente a recipeIngredients, normalizando las relaciones anteriores, a fin de evitar la dependencia y repetición de los datos.

Diseño y Gestión de Tablas

Tabla users: La tabla users contiene los registros que identifican y describen los usuarios del sistema. Cada registro usuario debe estar asociado a uno y solo un registro de la tabla Persona.

Atributos

- id: identificador secuencial asignado por el sistema.
- fullname: nombre completo del usuario que se registra.
- username: nombre de usuario para iniciar sesión.
- password: contraseña de usuario.

	TIPO DE DATO	NULO	DUPLICADO	
id	integer	NO	NO	Clave Primaria
fullname	Varchar (100)	NO	SI	

username	Varchar (50)	NO	NO	
password	Varchar (50)	NO	SI	

Tabla 2. Tabla users

Procedimientos

- addUser (fullname, username, password). Inserta un registro nuevo en la tabla users.

- getUserID (ID, userID). Carga dentro de la constante userID el resultado de la sentencia SELECT siguiente:

```
SELECT * FROM users WHERE id = ID;
```

- getUserUsername (pUsername, username). Carga dentro de la constante username el resultado de la sentencia SELECT siguiente:

```
SELECT * FROM users WHERE username = pUsername;
```

- LastRecipes. Devuelve las últimas recetas agregadas. Utilizando la siguiente sentencia:

```
SELECT * FROM recipeCate ORDER BY created_at DESC LIMIT 8;
```

Tabla favorites: Esta tabla está diseñada para almacenar y relacionar las recetas favoritas de cada usuario.

Atributos

- id: Identificador secuencial asignado por el sistema.
- user_id: clave para relacionar con la tabla users.
- recipe_id: clave para relacionar con la tabla recipes.

	TIPO DE DATO	NULO	DUPLICADO	
id	integer	NO	NO	Clave Primaria
user_id	integer	NO	SI	Clave Foránea
recipe_id	integer	NO	SI	Clave Foránea

Tabla 3. Tabla Favorites

Tabla recipes: La tabla recipes contiene la información general sobre cada receta agregada por cada usuario.

Atributos

- id: Identificador secuencial asignado por el sistema.
- title: Corresponde al titulo (nombre) de la receta.
- description: Descripcion de la receta.
- steps: Pason, preparación y cocción de la receta.
- created_at: Fecha en que se creó el registro de la receta.
- users_id: id del usuario que agregó la receta.
- category_id: id a la categoría correspondiente a la receta.

	TIPO DE DATO	NULO	DUPLICADO	
id	integer	NO	NO	Clave Primaria
title	Varchar (100)	NO	SI	
description	text	SI	SI	
steps	text	NO	SI	
create_at	timestamp	NO	SI	
users_id	integer	NO	SI	Clave Foránea
category_id	integer	NO	SI	Clave Foránea

Tabla 4. Tabla Recipes

Procedimientos

- addRecie(title, description, steps, category).Inserta un registro nuevo en la tabla.
- editRecipe(id). Permite editar la receta con id = id.
- deleteRecipe(id). Permite borrar la recete con id = id.

Tabla categories: En esta tabla se encuentran los registros de las categorías posibles que puede tener una receta.

Atributos

- id: Identificador secuencial.
- category: nombre de la categoria.

	TIPO DE DATO	NULO	DUPLICADO	
id	integer	NO	NO	Clave Primaria
category	Varchar (50)	NO	NO	

Tabla 5. Tabla Categories

Vistas

- recipeCate. Permite conocer la categoría asociada a una receta

de acuerdo a su id. Utilizando la siguiente sentencia:

```
SELECT r.title, r.description, r.created_at, r.steps, r.id, rc.category
FROM recipes r
JOIN categories rc on r.category_id = rc.id;
```

Tabla ingredients: Contiene un registro de todos los ingredientes posibles a utilizar en las recetas.

Atributos

- id: identificador secuencial.
- Name: nombre del ingrediente.

	TIPO DE DATO	NULO	DUPLICADO	
id	integer	NO	NO	Clave Primaria
name	Varchar (50)	NO	NO	

Tabla 6. Tabla ingredients

Tabla measures: Contiene los registros de las unidades para medir los ingredientes.

Atributos

- id: Identificador secuencial.
- measure: Nombre de la unidad de medida.

	TIPO DE DATO	NULO	DUPLICADO	
id	integer	NO	NO	Clave Primaria
measure	Varchar (50)	NO	NO	

Tabla 7. Tabla Measures

Tabla recipeIngredients: Permite relacionar la información general de cada receta agregada, con los ingredientes, cantidades y medidas a utilizar. Es gracias a esta tabla que se logra obtener la receta completa.

Atributos

- id: Identificador secuencial.
- amount: Cantidad de cada ingrediente a utilizar.
- ingredient_id: id de cada ingrediente de acuerdo a la receta agregada.

- recipe_id: id de la receta.
- measure_id: id de cada medida a utilizar.

	TIPO DE DATO	NULO	DUPLICADO	
id	integer	NO	NO	Clave Primaria
amount	Varchar (30)	SI	SI	
recipe_id	integer	NO	SI	Clave foránea
ingredient_id	integer	NO	SI	Clave foránea
measure_id	integer	SI	SI	Clave foránea

Tabla 8. Tabla recipeIngredients

Procedimientos

- addRecipeIngredient (recipe_id, ingredient_id. measure_id, amount). Inserta un registro nuevo en la tabla.

- EditRecipeIngredient (id). Edita una receta donde id = id.

- DeleteRecipeIngredient (id). Elimina una receta donde id = id.
- AllRecipes permite mostrar todas las recetas agregadas, utilizando la vista recetaCompleta.

Vistas

- recetaCompleta, permite mostrar la receta completa (titulo, descripciónn, ingredientes, medidas y pasos a seguir).

```
SELECT r.title AS 'Recipe', r.description, r.created_at, r.steps, ri.amount AS
'Amount', mu.measure AS 'Measure', i.name AS 'Ingredient'
FROM recipes r
JOIN recipeIngredients ri on r.id = ri.recipe_id
JOIN ingredients i on i.id = ri.ingredient_id
LEFT OUTER JOIN measures mu on mu.id = measure_id;
```

Triggers

- AGR_RECIPE: Se ejecuta antes de una operación de agregación o modificación de un registro en la tabla recipeIngredients. Este trigger realiza las acciones siguientes:

  1. Al insertarse un registro nuevo, asigna un valor secuencial al atributo ID.

  2. Lanza una excepción si el valor de código especificado para

measure esta vacío.

3. Lanza una excepción si el valor de código especificado para amount esta vacío

Sentencias SQL de las tablas y vistas

```sql
USE database_recipes;

CREATE TABLE users (
    id INT(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(16) NOT NULL,
    password VARCHAR(60) NOT NULL,
    fullname VARCHAR(100) NOT NULL
);

DESCRIBE users;

--CATEGORIES TABLE
CREATE TABLE categories (
    id INT(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
    category VARCHAR(30) NOT NULL
);

DESCRIBE categories;

--RECIPE TABLE


CREATE TABLE recipes (
    id INT(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(150) NOT NULL,
    user_id INT(11),
    category_id INT(11),
    description TEXT,
    steps VARCHAR(500),
    created_at timestamp NOT NULL DEFAULT current_timestamp,
    CONSTRAINT fk_user FOREIGN KEY  (user_id) REFERENCES users(id),
    CONSTRAINT fk_category FOREIGN KEY  (category_id) REFERENCES categorie
);
```

Figura 2. Creación de tablas users, categories y recipes

```sql
CREATE TABLE measures (
    id INT(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
    measure VARCHAR(30) NOT NULL UNIQUE
);

DESCRIBE measures;

CREATE TABLE ingredients (
    id INT(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL UNIQUE
);

DESCRIBE ingredients;


CREATE TABLE recipeIngredients (
    id INT(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
    measure_id INT(11),
    ingredient_id INT(11),
    recipe_id INT(11),
    amount VARCHAR(50),
    CONSTRAINT fk_measure FOREIGN KEY  (measure_id) REFERENCES measures(id),
    CONSTRAINT fk_ingredient FOREIGN KEY  (ingredient_id) REFERENCES ingredients(id),
    CONSTRAINT fk_recipe2 FOREIGN KEY  (recipe_id) REFERENCES recipes(id)
);
```

Figura 3. Creacion de tablas meaures, ingredientes y recipeIngredients.

```sql
CREATE VIEW recipeComplete AS
SELECT r.title AS 'Recipe', r.description, r.created_at, r.steps, ri.amount AS 'Amount'
FROM recipes r
JOIN recipeIngredients ri on r.id = ri.recipe_id
JOIN ingredients i on i.id = ri.ingredient_id
LEFT OUTER JOIN measures mu on mu.id = measure_id;



CREATE VIEW recipeCate AS
SELECT r.title, r.description, r.created_at, r.steps, r.id, rc.category
FROM recipes r
JOIN categories rc on r.category_id = rc.id;
```

Figura 4. creación de vistas

DEFINICIÓN DE INTERFACES

De acuerdo con la metodología se procedió a implementar los componentes funcionales de la aplicación, éste proceso permite definir las interfaces de usuario en lenguaje HTML en conjunto de frameworks y herramientas como Bootrstrap de CSS, para conseguir una interfaz mucho más amigable y mejorar la experiencia de usuario.

Pagina de Inicio: En está sección el usuario puede registrar y/o iniciar sesión.

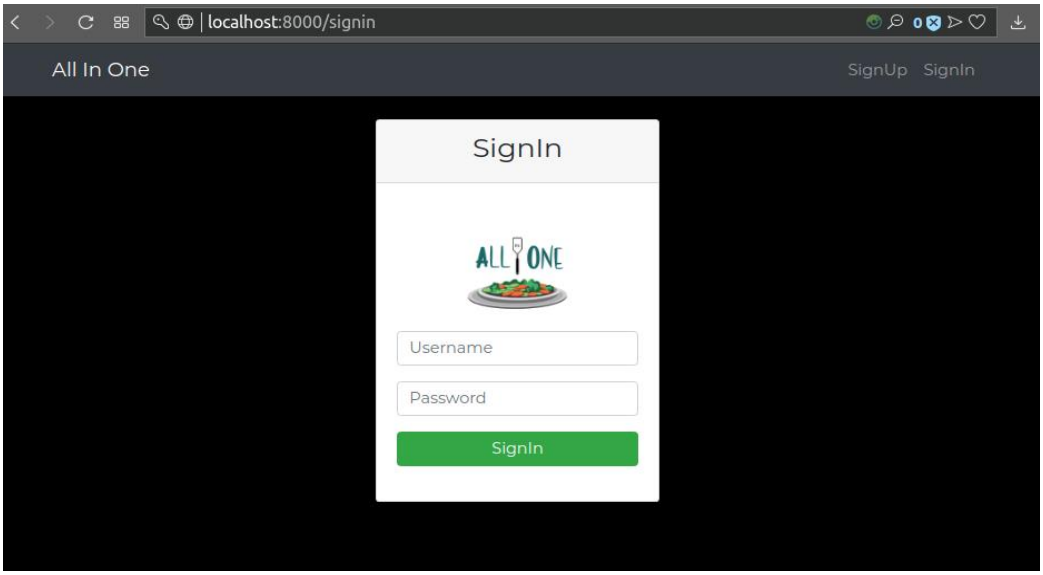

Figura 5. Inicio de Sesión

Principal: Se pueden ver las últimas recetas agregadas a la plataforma

(por cualquier usuario) junto a su categoría correspondiente. De igual manera permite ver todas las recetas existentes.

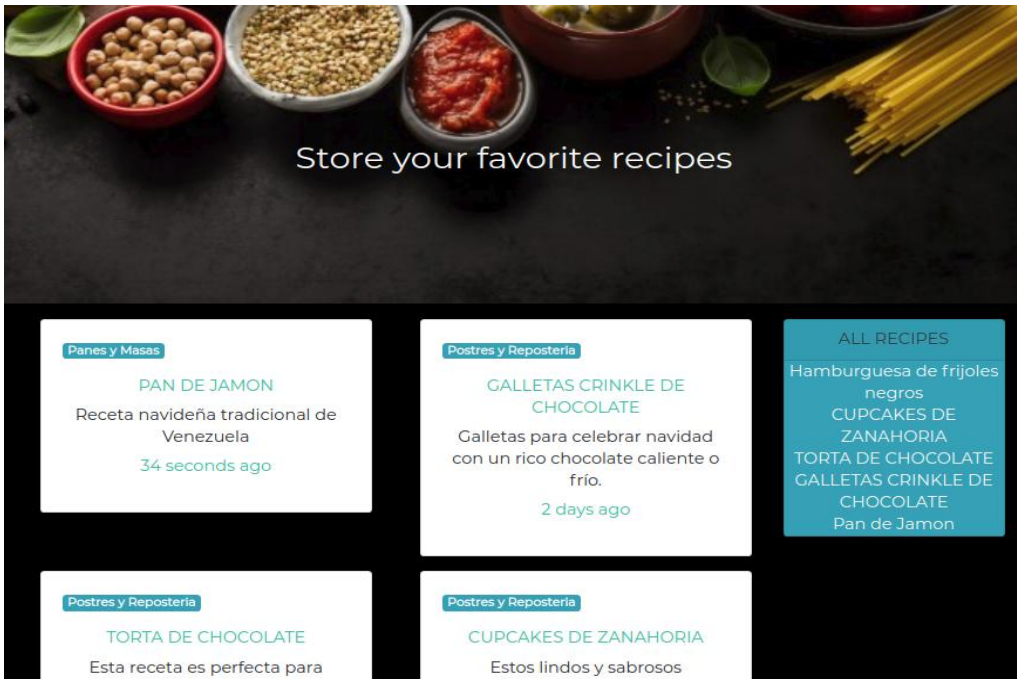

Figura 6. Página principal.

Perfil del Usuario: Se muestran las últimas recetas agregadas por el usuario. Aquí el usuario tiene la posibilidad de ver lo pasos, editar o eliminar una receta.

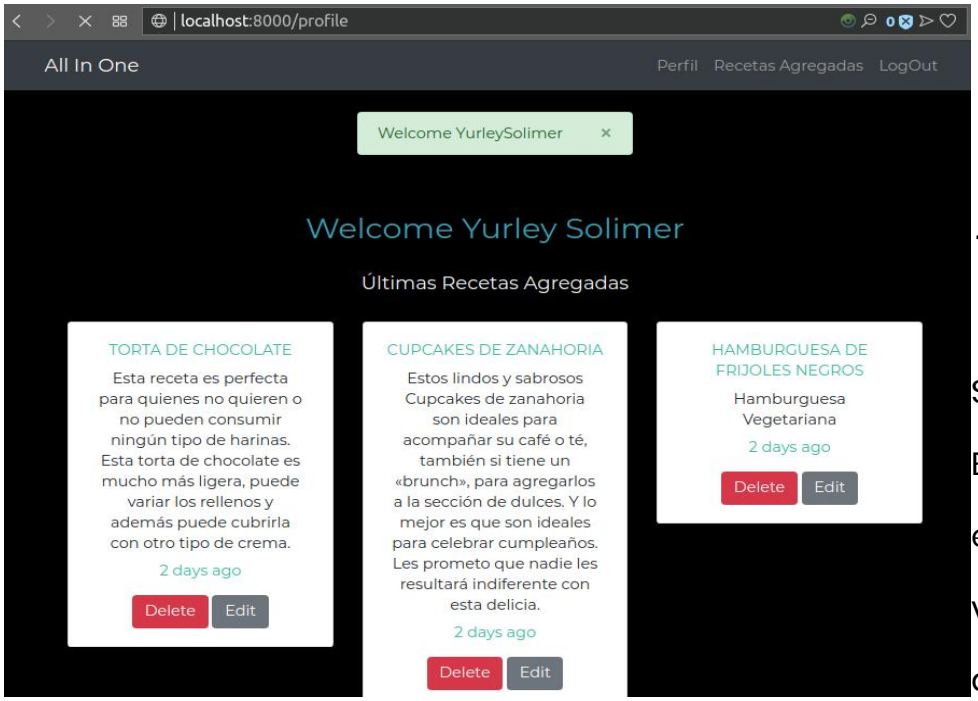

Figura 7. Perfil

Steps: En en esta vista donde el usuario puede leer los ingredientes y preparación de la receta.

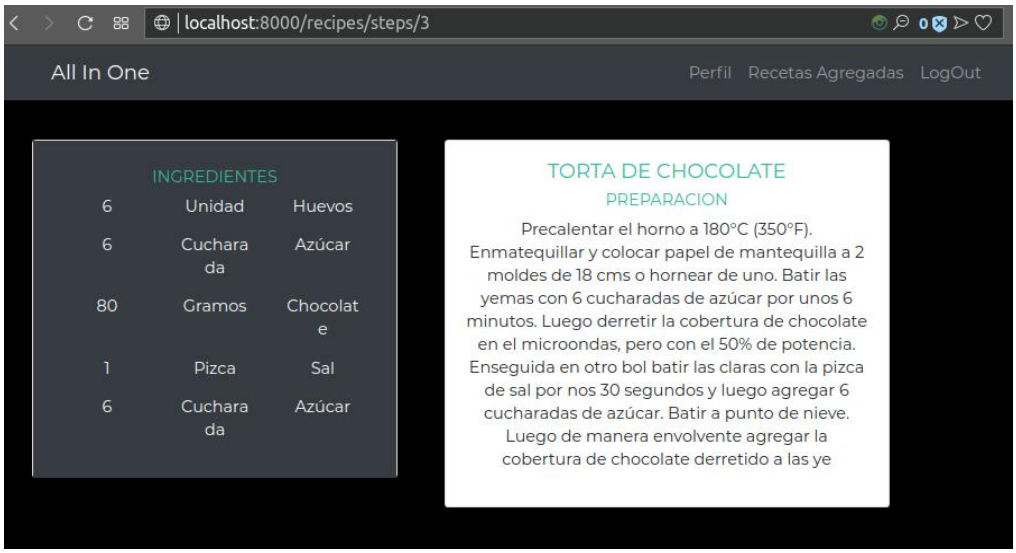

Figura 8. Preparación

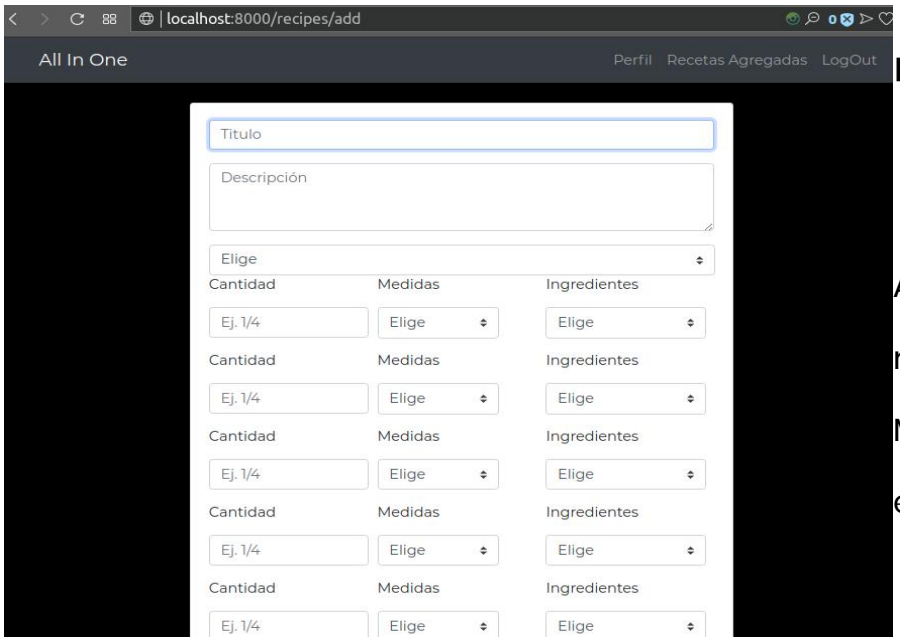

Figura 9. Agregar receta Agregar receta: Muestra el formulario donde el usuario podrá escribir la información necesaria para agregar una receta.

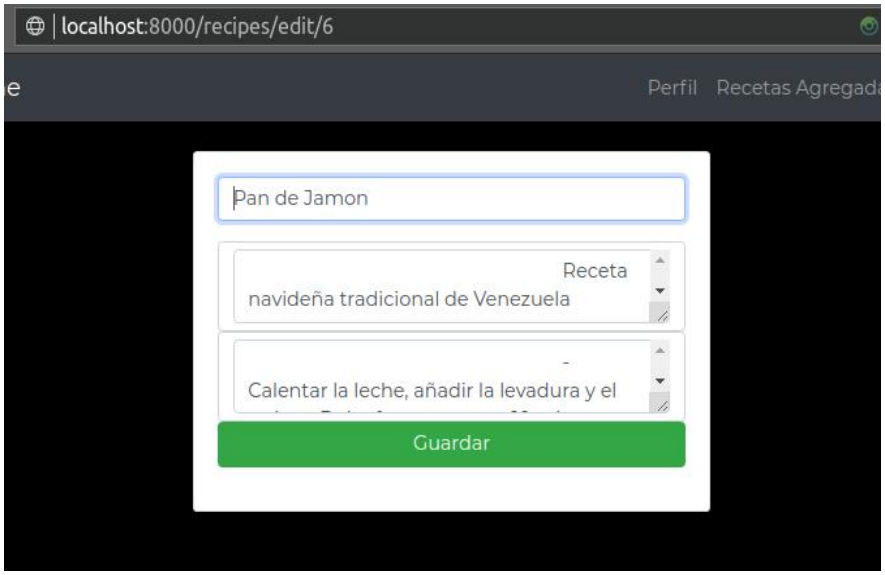

Figura 10. Editar receta

Editar receta: En esta vista el usuario tiene la posibilidad de editar sus recetas. P Para ello se muestra nuevamente un formulario con la información guardada previamente.

ESTRUCTURA DEL PROYECTO

La estructura de un proyecto en Node JS es un modelo empleado para todo proyecto, donde se encuentren los módulos y recursos a utilizar.

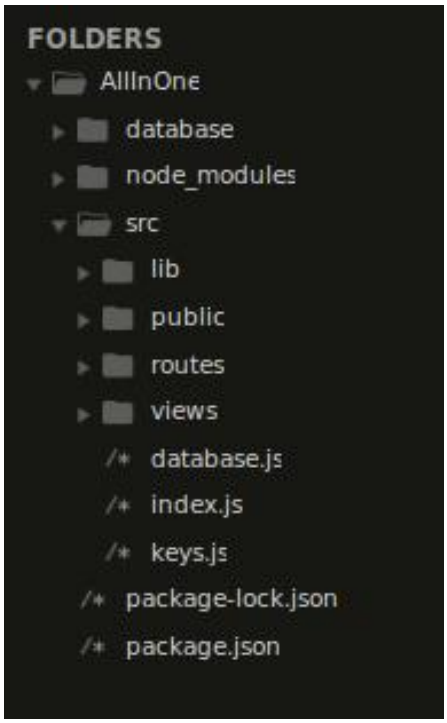

Figura 11. Estructura General

Dentro del proyecto AllInOne se encuentran las diversas carpetas y archivos principales del proyecto.

- database.js: Permite hacer la conexión de MySQL con el servidor creado por NodeJS.

- Index.js: Es el archivo principal, en el cual se exportan los modulos necesarios, se declaran las variables globlales, se crean las rutan principales y se configura el servidor para poder ejecutar la aplicación.
- Keys.js: Permite guardar de manera separada la información del usuario y clave del administrador de MySQL.
- Los paquetes de JSON contienen la configuración correspondiente a los modulos utilizados por NodeJS.

Por otra parte, la carpeta database almacena los archivos relacionados a la base de datos con las principales sentencias para crear las tablas e insersión de los datos.

En cuanto node_modules contiene todos los módulos de Node JS necesarios para que la aplicación pueda funcionar.

Finalmente es dentro de src donde se encuentra lo relacionado al código funcional y al diseño de la interfaz web.

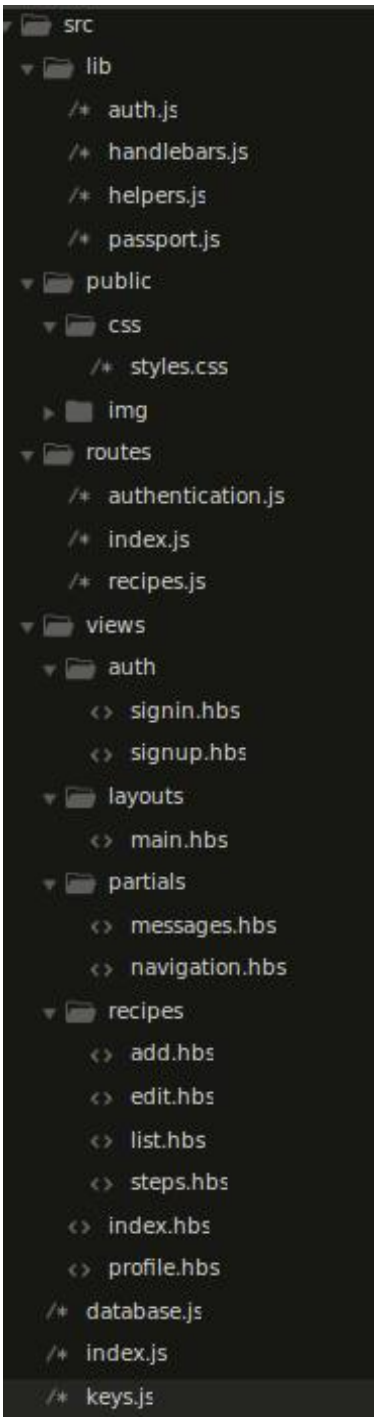

Figura 12. Estructura carpeta src

- Lib contiene los documentos relacionado a la autenticacion de los usuarios que se registran en la plataforma.

  - El archivo auth.js contiene las funciones que permiten saber si un usuario está logueado o no, con esta información se da permiso al usuario para realizar las operaciones básicas de agregar, editar y eliminar.

  - Handlebars contiene funciones para convertir el formato del tiempo recibido al crear un nuevo registro.

  - Helpers.js permite realizar el encriptado de las contraseñas.

  - Passport.js permite validar la contraseñ ingresada por el usuario.

- Dentro de Public se encuentran los archivos de CSS relacionados al estilo y diseño de la aplicación.

- La carpeta routes es la de mayor importancia, es en ella donde se encuentran los archivos que permiten enviar peticiones a la base de datos y gestionar la información recibida.

  - authenticacion.js nos permite conocer el nombre y id del usuario registrado, para redirigirlo a su perfil y poder activarlo en la plataforma.

- index.js gestiona lo relacionado a la página principal de la aplicación, como mostrar las últimas recetas, mostrar todas las recetas y mostrar las categorías de cada receta.
- recipes.js contiene la mayor parte del código del CRUD, es aquí donde se encuentran las funciones para agregar, ver, editar y eliminar las recetas, de acuerdo a cada usuario autorizado.

```
router.get('/add', isLoggedIn, async (req, res) => {
    const measure = await pool.query('SELECT * FROM measures');
    const ingredient = await pool.query('SELECT * FROM ingredients');
    const category = await pool.query('SELECT * FROM categories');
    res.render('recipes/add.hbs', {measure, ingredient, category});

});


router.get('/steps/:id', isLoggedIn, async (req, res) => {
    const { id } = req.params;
    const recipes = await pool.query('SELECT * FROM recipes WHERE id
    const recipeComplete = await pool.query('SELECT * FROM recipeComp

    res.render('recipes/steps.hbs', {recipeComplete, recipes});

});


router.post('/add', isLoggedIn, async (req, res) => {
    const {category} = req.body;
    const category_id = await pool.query('SELECT id FROM categories W
    console.log(category_id);

    const {title, description, steps} = req.body;
    const newRecipe = {
        title,
        category_id : category_id[0].id,
```

Figura 13. Peticiones para agregar receta

Figura 14. Peticiones para editar y eliminar.

```
router.get('/delete/:id', isLoggedIn, async (req, res) => {
    const { id } = req.params;

    await pool.query('DELETE FROM recipeIngredients WHERE recipe_id = ?',
    await pool.query('DELETE FROM recipes WHERE ID = ?', [id]); //Elimina

    req.flash('success', 'Recipe Deleted Successfully');
    res.redirect('/recipes');
});

router.get('/edit/:id', isLoggedIn, async (req, res) => {
    const { id } = req.params;
    const recipes = await pool.query('SELECT * FROM recipes WHERE id = ?'
    res.render('recipes/edit.hbs', {recipe: recipes[0]});
});

router.post('/edit/:id', isLoggedIn, async (req, res) => {
    const { id } = req.params;
    const { title, description, steps } = req.body;
    const newRecipe = {
        title,
        description,
        steps
    };
```

- Por último la carpeta views contiene el cuerpo HTML de cada una de las vistas utilizadas en la aplicación.

CONCLUSIONES

Para utilizar cualquier tecnología para bases de datos es necesario entender el diseño y modelado requerido a fin de lograr tener la mejor arquitectura posible. El proyecto permitió consolidar los conocimientos de la materia Base de Datos, permitiendo crear un modelo propio de bases de datos e implementar el diseño conceptual y lógico de la aplicación.

La creación de un buen diseño en conjunto con procedimientos y vistas permiten que el acceso a la base de datos sea mucho más rápido, logrando así aplicaciones más eficientes y de mejor calidad.

Se logró crear un CRUD utilizando las tecnologías más actuales de JavaScript, ejecutando un servidor con NodeJS e implementando diferentes módulos en conjunto con Express. Este CRUD permite a usuarios crear, actualizar y eliminar diferentes recetas desde su perfil de All In One, permitiéndoles una mejor gestión de la información que quieren mantener.

ANEXOS

Repositorio; https://github.com/YurleySolimer/Databases-Project