# Recetas FIUBA


| | |
|---|---|
| Bermudez, Agustin | Padrón: 111863, abermudez@fi.uba.ar |
| Calderón, Tiago André | Padrón: 111894, tcalderon@fi.uba.ar |
| Gonzalez Pautaso, Mateo | Padrón: 111699, magonzalezp@fi.uba.ar |
| Mendez, Alejandro Oscar | Padrón: 111756, aomendez@fi.uba.ar |
| Urbina, Cristian Ezequiel | Padrón: 112374, curbina@fi.uba.ar |
| Vega, Lorenzo Julián | Padrón: 111911, ljvega@fi.uba.ar |
| Verruno, Ignacio | Padrón: 111910, iverruno@fi.uba.ar |


June 17, 2024

## Resumen


Recetas FIUBA es un proyecto diseñado para que los usuarios puedan elegir recetas mediante una página web y que esta misma le indique los ingredientes necesarios para realizarla.

La página web permite que los usuarios no solo elijan y visualicen las recetas, sino también poder iniciar sesión para sugerir las recetas que gusten. Estas recetas sugeridas serán cargadas a la base de datos con el nombre del usuario dueño de la receta.

La base de datos es MySQL de tipo relacional y va a contener dos tablas. La principal contiene los nombres de las recetas, un enlace a una imagen, un JSON que contiene los ingredientes y sus cantidades y el usuario que sugirió la receta. La segunda tabla contiene los nombres de usuarios y sus contraseñas encriptadas.

Las consultas de datos se realizaran mediante una API implementada en Flask, donde cada endpoint contiene una query particular para cumplir una necesidad del frontend.


## Palabras clave


- Python
- Flask
- Base de datos
- mySQL
- Recetas
- Ingredientes

- Inicio de sesión
- Listado de compra
- APIs
- Frontend
- Backend
- Interacción
- Encriptación
- JSON
- Cantidad de ingredientes
- Endpoint
- Consulta
- Clave foránea

## Abstract

Recetas FIUBA is a project designed for users to choose recipes through a website, which will also indicate the necessary ingredients to make them. The website allows users not only to select and view recipes but also to log in to suggest recipes they like. These suggested recipes will be added to the database with the name of the user who owns the recipe.
The database is a relational MySQL database and will contain two tables. The main table contains the names of the recipes, a link to an image, a JSON that includes the ingredients and their quantities, and the user who suggested the recipe. The second table contains the usernames and their encrypted passwords. Data queries will be made through an API implemented in Flask, where each endpoint contains a particular query to meet the needs of the frontend.

## Keywords

- Python
- Flask
- Database
- mySQL
- Recipes
- Ingredients
- Login
- Shopping list
- APIs
- Frontend
- Backend
- Interaction
- Encryption
- JSON
- Ingredient Quantities
- Endpoint
- Query
- Foreign key

# Introducción

- Breve descripción de este documento:

    1. Título del proyecto
    2. Listado de integrantes
    3. Resumen del proyecto
    4. Palabras clave relacionadas con el proyecto
    5. Proyect summary
    6. Proyect keywords
    7. Introducción al proyecto (Índice del documento y problema que plantea resolver)
    8. Solución propuesta (Incluye el recorrido de la página desde la perspectiva del usuario y también el flujo del programa, es decir, como funciona)
    9. Pruebas de funcionamiento (Documentación de los ensayos que se fueron realizando en el programa)
    10. Plan de actividades (Documentación del paso a paso del desarrollo con fechas incluidas)
    11. Hipótesis y supuestos (Circunstancias supuestas por el equipo para comprender el motivo y desarrollo del programa)
    12. Referencias (Bibliografía de recursos utilizados en el desarrollo)

## Problema a resolver

Mejorar la organización y facilitar la tarea de los usuarios a la hora de hacer una lista de la compra con ingredientes necesarios para cocinar en su día a día. También lograr adaptarse a los gustos gastronómicos de cada usuario particular mediante una oferta de recetas varias y también un apartado para incluir las recetas favoritas de cada uno.

# Solución propuesta

Para cumplir este objetivo planteamos la creación de una página web donde los usuarios podrán acceder a una herramienta que crea las listas de la compra de ingredientes a partir de las recetas que gusten cocinar. Para adaptarse a los gustos de todos los usuarios, contamos con un apartado donde cada uno podrá agregar sus recetas favoritas para que disfruten el resto de usuarios, abarcando la mayor cantidad de estilos de cocina.

## Historia de usuario

Dentro de la vista principal de la página web se encontrará una pequeña introducción a la página, con apartados artísticos y también explicativos de las funcionalidades.
La página cuenta con una barra de navegación para facilitar el acceso a los demás apartados de la página web:

- Menú: En este apartado se muestran todas las recetas que están presentes en la base de datos de la página web, mostrando una imagen de cada una y también una pequeña descripción del plato
- Lista de la compra: En esta vista se presentan checkbox con todas las recetas con las que cuenta la página. Cuando el usuario selecciona en las que está interesado, aprieta un botón que lo redirige a una vista con los ingredientes necesarios y sus respectivas cantidades.
- Sugerencias: Se presenta un inicio de sesión. Al ingresar un conjunto de credenciales válidas, se presenta una vista que le pide al usuario datos generales de la receta. Al completarlos se presenta una nueva vista donde el usuario ingresa los ingredientes, sus cantidades y la unidad de medida y también un enlace a una imagen de referencia. Una vez completado esto, la página web ya cuenta con esta receta.

## Flujo del programa

El frontend está implementado usando Python Flask, todas las vistas a las que el usuario puede acceder están definidas en sus respectivos endpoints y renderizando archivos HTML.
Los HTML utilizaran CSS, JavaScript. Para mejorar su apariencia y legibilidad.
Se cuenta con una base de datos mySQL, implementada en un contenedor de Docker. Dentro de la base de datos se encontrarán dos tablas. La primera consta de la tabla de recetas, donde se almacenan todos los datos relevantes de la misma. La segunda tabla, guarda los usuarios y sus contraseñas encriptadas, solo los usuarios presentes en esta tabla están autorizados a sugerir nuevas recetas. Dentro de la tabla de recetas existe una columna que respecta al usuario dueño de la misma, este apartado es una clave foránea de la segunda tabla (los nombres de usuario).
Para llevar a cabo la recolección de datos desde la base mySQL se implementó una API en Python Flask. La misma cuenta con todos los endpoints necesarios para el correcto funcionamiento de la página web y también cuenta con algunos endpoint demostrativos que no se implementan en los apartados de esta última.
Tanto el frontend como el backend cuentan con su respectiva documentación. Explicando el código mediante comentarios en los sectores donde lo consideramos necesario.

En el repositorio también existen dos scripts escritos en bash. Uno de estos facilita la instalación de todas las dependencias que necesita el frontend. El segundo instala todo lo necesario para poder levantar un contenedor en docker con la base de datos y también las dependencias que necesita la API para conectarse a la misma.

# Pruebas y/o validación

- Para las pruebas de funcionamiento de la API, se usó la aplicación Postman, donde se mandan solicitudes de prueba para testear las respuestas de la API.
- Para verificar el correcto funcionamiento del login se probó ingresar con credenciales inválidas y también acceder directamente desde el URL, tratando de hacer un bypass al login.
- Previo a implementar una columna de imágenes en la tabla de recetas, se usó una imagen dummy para todas las recetas. Al verificar que la imagen dummy se mostraba correctamente añadimos la consulta de las imágenes al endpoint particular que usa el apartado de menú.
- Al momento de la implementación del apartado que crea la lista de la compra notamos que debíamos forzar que el usuario seleccione al menos dos checkbox (función de JavaScript). Esto porque las checkbox seleccionadas se guardan en una tupla que se envía a la api. El problema es que la sintaxis de Python de tuplas de un solo elemento es incompatible con la sintaxis mySQL.
- En todos los campos presentes en el apartado de sugerir los HTML tienen campos con la propiedad 'Required' para evitar conflictos con campos vacíos en los formularios.
- Al final de la implementación del proyecto. Se clonó el repositorio y ejecutaron los scripts en una máquina virtual nueva. Así validando que las dependencias sean instaladas correctamente.

# Plan de actividades

En el proyecto nos basamos en el uso de un tablero Kanban, con la aplicación Trello, donde hay cartas que representan tareas, con un ID único para ser vinculadas a los commits realizados en el repositorio de Github. Por cada commit, siempre va a haber un CARD ID relacionado a este. En la aplicación de trello, también se pueden ver las fechas donde cada una de las cartas cambiaba de estado.
En el repositorio de GitHub hay distintas ramas para seguir el flujo de trabajo de las actividades planteadas, una vez que se termina con dicha tarea, se crea una Pull Request para solicitar unir la rama actual con la main del proyecto. Esta Pull Request debe ser aceptada por un integrante del equipo que no sea autor de la misma.

Una vez que las Pull Request sean aceptadas e incluidas en la rama main, se podrá ver reflejado el autor de la misma y también la fecha en la que el cambio se oficializó. Cada vez que esto sucede queda en evidencia una versión del proyecto con las últimas características añadidas.
Para la comunicación y organización del equipo, se usa un espacio de trabajo en Slack incluyendo a integrantes, donde periódicamente se realizan llamadas para coordinar el trabajo.

# Entregables

- Creación de repositorio en GitHub, listado de tareas en tablero kanban, mockup de las vistas de la página y creación del espacio de trabajo.
- Vistas HTML, adaptación del template a Python Flask y avances en el desarrollo de la API. Modificación de los archivos HTML para cumplir las necesidades del proyecto.
- Demostración de la mayor parte de la funcionalidad de la página web.
- Entrega final. Desarrollo completo del proyecto con todas sus características funcionales

# Hitos

- Creación del repositorio en GitHub y primeros commits
- Creación del logo de la página.
- Escritura del archivo README.md en el repositorio de GitHub
- Template de página web seleccionado
- Adaptación del template a Flask, modularización del código HTML con estructura block. Todos los datos presentes en las vistas están hardcodeados en los HTML.
- Creación del contenedor docker donde esta la base de datos del proyecto
- Creación del archivo api.py, donde están presentes todos los endpoints que interactúan con la base de datos y le envían las respuesta al frontend.
- Verificaciones de funcionamiento tanto del frontend como del backend, por separado.
- Creación del login de usuarios.
- Encriptación de las contraseñas de los usuarios.

- Se eliminan los datos hardcodeados de los HTML por datos extraídos desde la base de datos, usando la API.
- Completar la documentación del proyecto mediante el uso de Overleaf.
- Lograr hostear el frontend, la api y su base de datos en la nube. Python Anywhere

## Hipótesis y Supuestos

Para el proyecto ”Recetas FIUBA”, partimos de algunas hipótesis y supuestos clave que nos guiarán durante el transcurso del desarrollo del proyecto. Creemos que facilitar la creación de listas de compras basadas en las recetas seleccionadas debería mejorar la organización y eficiencia de los usuarios al hacer las compras. También esperamos que permitirles sugerir nuevas recetas renueve continuamente nuestra base de datos, manteniendo el interés de la comunidad.
Suponemos que nuestra infraestructura podrá manejar el volumen de usuarios esperado sin problemas de rendimiento. También asumimos que la plataforma podrá escalar según sea necesario para soportar más usuarios y datos en el futuro.
Para la implementación de la página web en este proyecto se usará una template gratuita obtenida desde intenet.
El apartado de inicio de sesión será aplicado con usuarios que fueron previamente almacenados en la base de datos. Los procesos de registro de nuevos usuarios quedan fuera del alcance del desarrollo.
En la API se cuentan con algunos endpoints que no se conectan con el frontend del proyecto. Estos están a modo de demostración para usar los verbos HTTP que no son necesarios para cumplir con el alcance del proyecto. También existen algunos endpoint relacionados a la manipulación de credenciales de los usuarios.

## Referencias

- Template de página web usada en el desarrollo de este proyecto: Savory Free Website Template for Restaurants Websites
- Las imágenes usadas en la página web fueron extraídas desde la página: Freepik
- Pagina donde esta hosteda la pagina, la api y la base de datos Python Anywhere

## Anexos

Al día de la fecha no se necesitó implementar ningún anexo para la documentación de este proyecto.