PROYECTO FINAL

Ingeniería de Software I

Wullfredo Javier Barco Godoy - wbarco@unal.edu.co
Brahian Camilo Gómez Carvajal - bgomezca@unal.edu.co
Juan David Rivera Buitrago - jriverabu@unal.edu.co
William Darío Vanegas Marín - wvanegas@unal.edu.co

Profesor
Oscar Eduardo Alvarez Rodríguez

Universidad Nacional de Colombia
Facultad de Ingeniería
Ingeniería de Sistemas y Computación
Bogotá, Colombia
09 de Febrero de 2025

1. Pre Requerimientos:

Están en el archivo README.md que está en el repositorio de nuestro proyecto, sin embargo aquí se resume lo más elemental de dicho archivo:

Nombre del Proyecto: ChazaUNApp

Descripción del Proyecto: ChazaUNApp es una plataforma pensada y diseñada para la gestión y promoción de pequeños negocios y ventas informales dentro de la Universidad Nacional de Colombia. Inicialmente, permite a los dueños de las chazas o chazeros registrar sus negocios y poder hacer gestión de los estudiantes que serán vendedores o trabajadores de las chazas, así como también le permite a los estudiantes poder vincularse para trabajar en alguna de las chazas. Posteriormente, se espera poder vincular a los compradores para que a través de esta aplicación puedan encontrar productos y servicios en alguna de las chazas disponibles.

Tecnologías utilizadas:

- **Lenguaje de Programación:** Dart.
- **Frameworks:** Flutter.
- **Servicios externos:** Firebase (Autenticación, Base de datos en tiempo real), Google Maps API.

Ícono o Logo del Proyecto:

Imagen 1. Logo del Proyecto.

2. Levantamiento de Requerimientos

Origen de la Idea

La idea de desarrollar **ChazaUNApp** surgió de la observación y análisis de la dinámica de comercio informal y de trabajo dentro del campus de la sede Bogotá de la Universidad Nacional de Colombia. Muchos estudiantes buscan conseguir algún empleo que les permita poder

obtener ingresos económicos sin descuidar sus responsabilidades académicas, a la vez que los dueños de las chazas o “chazeros” no cuentan con una herramienta que les permita poder establecer un contacto más cercano con estudiantes que puedan trabajar en algunos intervalos de tiempo dentro del horario del funcionamiento de las chazas. Además se ha observado que en los espacios informales o chazas se realiza la venta de productos y servicios sin contar con una plataforma que facilite su visibilidad y gestión. A través de entrevistas hechas a compradores y a trabajadores y dueños de las chazas quienes ejercen el rol de vendedores, identificamos que existe una necesidad clara de digitalizar y organizar este tipo de comercio para hacerlo más accesible y eficiente.

La principal problemática que se busca resolver con ChazaUNApp es la falta de un canal de comunicación eficiente entre los estudiantes que buscan ser trabajadores de alguna de las chazas y los chazeros, en comunión con los estudiantes y demás miembros de la comunidad universitaria que desean acceder a la información más relevante de las chazas existentes en la universidad de manera rápida y confiable.

Proceso de Selección de la Idea

Para la elección del proyecto, el equipo realizó una sesión de brainstorming en la que se propusieron varias ideas. Posteriormente, se evaluaron en función de su viabilidad técnica, impacto en la comunidad y alineación con nuestras habilidades y expectativas de aprendizaje. A través de votaciones y debate entre todos los integrantes del grupo, se concluyó que **ChazaUNApp** era una solución innovadora y con alto potencial de utilidad dentro del entorno universitario. Se asignaron roles iniciales para la investigación y planificación del proyecto, buscando con ello garantizar un trabajo coordinado y eficiente.

Problemáticas que Busca Resolver

Las principales problemáticas identificadas son las siguientes:

- **Falta de visibilidad de ofertas laborales**: Los estudiantes que deseen y/o necesiten trabajar no encuentran una plataforma concreta donde se encuentren fácilmente las chazas en las cuales haya vacantes disponibles.
- **Dificultad en la selección de candidatos:** Los chazeros tienen problemas para filtrar y seleccionar a los candidatos adecuados.
- **Comunicación ineficiente:** La coordinación entre chazeros y los estudiantes postulantes es deficiente debido a la falta de un canal de comunicación claro y directo.
- **Gestión inadecuada de horarios:** Problemas para coordinar horarios de entrevistas y de trabajo.
- **Falta de visibilidad**: Los dueños y trabajadores de las chazas no cuentan con una plataforma digital donde promocionar sus productos o servicios.
- **Dificultad en la localización**: Hasta el momento no existe un medio unificado para conocer la ubicación y disponibilidad de las chazas.

- **Falta de confianza y seguridad**: Los compradores potenciales, es decir, los miembros de la comunidad universitaria no cuentan con referencias ni reseñas sobre los vendedores, lo que puede generar incertidumbre al momento de realizar compras.
- **Falta de organización**: Hasta ahora no existe un sistema centralizado que permita una mejor gestión de la oferta y demanda.

Expectativas de los Usuarios Potenciales

A través de una consulta de opinión sobre una propuesta de aplicación como la que se ha venido planteando en este documento a otros compañeros, compañeras y demás miembros de la comunidad universitaria, se destaca y se sintetiza que los usuarios potenciales de **ChazaUNApp** esperan:

- Lograr una comunicación más rápida y eficiente entre chazeros y estudiantes postulantes.
- Encontrar y seleccionar fácilmente las chazas para poder aplicar a trabajar en alguna de ellas.
- Encontrar fácilmente las chazas así como los productos y servicios dentro de la universidad.
- Conocer la ubicación de las chazas y los horarios de atención de cada una de ellas.
- Acceder a reseñas y calificaciones de otros usuarios para tomar decisiones informadas.
- Tener una interfaz sencilla e intuitiva que facilite la navegación.
- Contar con notificaciones y actualizaciones sobre el estado de las postulaciones, así como con la disponibilidad de productos.

Beneficios Esperados del Proyecto

Para todos los integrantes del equipo, el desarrollo de **ChazaUNApp** representa en conjunto una oportunidad de crecimiento en diversas áreas, tales como:

- Aplicación de conocimientos en desarrollo móvil y uso de **Flutter**.
- Experiencia en gestión de proyectos de software en un entorno real.
- Aprendizaje sobre integración de bases de datos en la nube mediante **Firebase**.
- Desarrollo de habilidades de UX/UI para la creación de una interfaz amigable.
- Impacto positivo en la comunidad universitaria al proporcionar una solución digital innovadora.

Funcionalidades Identificadas

Con base en las problemáticas mencionadas, se han identificado las siguientes funcionalidades clave para la aplicación:

- Registro y autenticación de usuarios (estudiantes trabajadores, estudiantes postulantes y chazeros).
- Creación y gestión de perfiles de usuarios (estudiantes trabajadores, estudiantes postulantes y chazeros).

- Gestión de horarios.
- Sistema de búsqueda y filtrado de candidatos.
- Búsqueda y consulta de chazas disponibles.
- Sistema de mensajería integrada entre estudiantes postulantes y chazeros.
- Geolocalización de las chazas en un mapa interactivo.
- Historial de postulaciones.
- Sistema de búsqueda y filtrado de productos/servicios.
- Sistema de pedidos en línea.
- Calificación y reseñas entre usuarios.
- Notificaciones sobre disponibilidad y promociones.

3. Análisis de Requerimientos

Número	Funcionalidad	Prioridad (MoSCoW)	Esfuerzo (Fibonacci)	Justificación
1	Registro y autenticación de usuarios	Must	5	Es una funcionalidad crítica para el acceso al sistema ya que requiere integración con Firebase Authentication, configuración de seguridad y pruebas de autenticación.
2	Creación y gestión de perfiles de usuarios	Must	8	Implica la creación de perfiles con opciones de personalización, subida de imágenes y edición de datos, lo que aumenta su complejidad técnica.
3	Gestión de horarios	Should	8	Implica la creación de perfiles con opciones de personalización, subida de imágenes y edición de datos, lo que aumenta su complejidad técnica.
4	Sistema de búsqueda y filtrado de candidatos	Must	8	Esta funcionalidad mejora la organización del proceso de

				contratación, permitiendo tanto a chazeros como a estudiantes postulantes coordinar mejor sus tiempos para evitar al máximo posibles conflictos de horarios.
5	Búsqueda y consulta de chazas disponibles	Must	8	Esta funcionalidad crítica mejora la experiencia del usuario, ya que permite encontrar fácilmente las chazas existentes.
6	Sistema de mensajería integrada entre estudiantes postulantes y chazeros	Should	13	Implica que puede haber una comunicación más directa entre estudiantes postulantes y chazeros. Sin embargo, no es una funcionalidad crítica.
7	Geolocalización de las chazas en un mapa interactivo	Should	13	La implementación de Google Maps API requiere permisos de ubicación, bases de datos para almacenar coordenadas y una UI dinámica, lo que implica una mayor complejidad.
8	Historial de postulaciones	Could	13	Por medio de esta funcionalidad, los estudiantes postulantes pueden ver un registro de todas sus aplicaciones anteriores. Es una de las funcionalidades con menor prioridad.

9	Sistema de búsqueda y filtrado de productos/servicios	Should	8	La búsqueda eficiente es importante aunque inicialmente no es algo de mucha prioridad. Su desarrollo implica bases de datos indexadas y optimización de consultas para mejorar la experiencia del usuario.
10	Sistema de pedidos en línea	Should	13	A través de esta funcionalidad los usuarios compradores pueden realizar pedidos a través de la aplicación de los productos de una chaza previamente seleccionada. Implica bases de datos indexadas y optimización de consultas además de una actualización constante para mejorar la experiencia de usuario.
11	Calificación y reseñas entre usuarios	Should	5	Aunque no es esencial para el funcionamiento inicial, mejora la confiabilidad de la plataforma. Requiere la implementación de una base de datos y lógica de manejo de reseñas.
12	Notificaciones sobre disponibilidad y promociones	Could	8	Una funcionalidad útil pero no esencial en la primera versión. Su implementación con Firebase Cloud Messaging es factible,

				pero podría añadirse en una fase posterior.

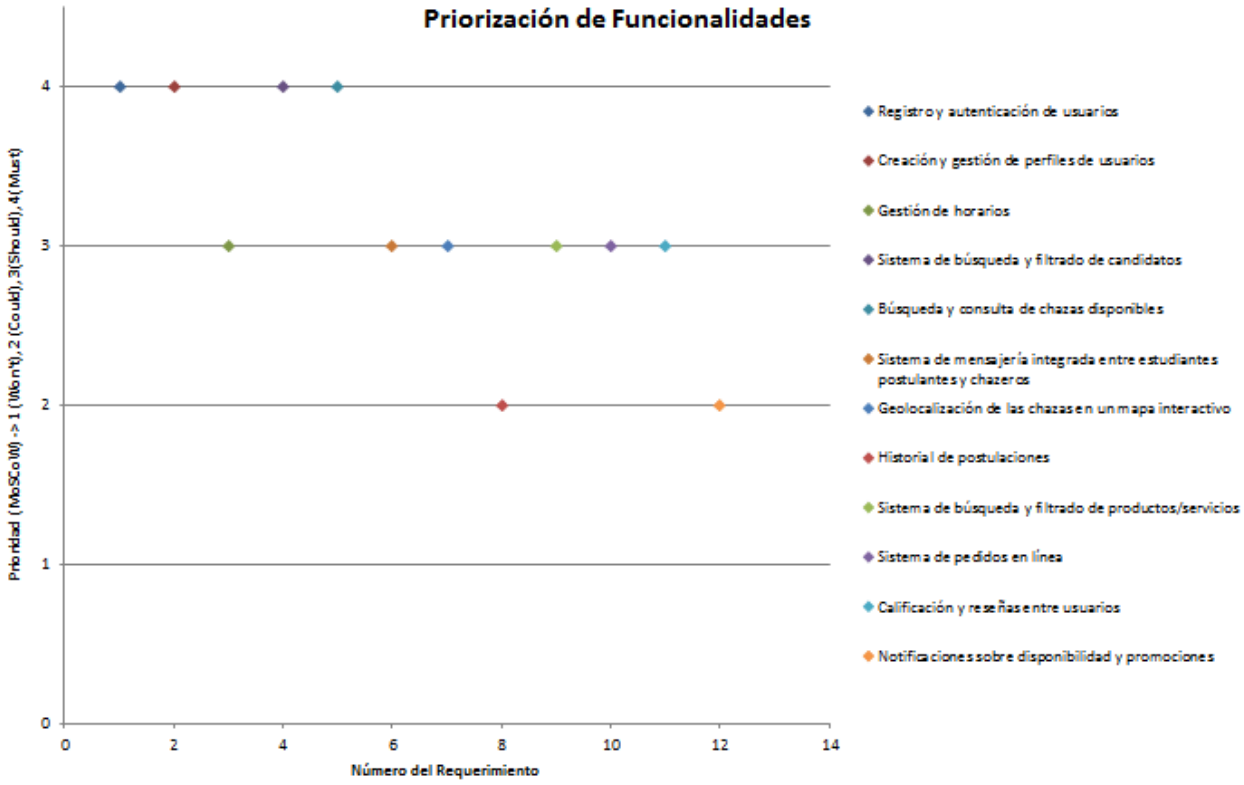

Imagen 2. Gráfica de Priorización de Funcionalidades. Esta gráfica está realizada con base en el análisis de requerimientos hecho previamente. El eje horizontal representa el número del requerimiento obtenido del cuadro de análisis de requerimientos, y el eje vertical representa las categorías del método de Moscow.

4. Análisis de Gestión de Software

Tiempo

Se estima que el desarrollo del proyecto tome aproximadamente **4 meses**, los cuáles se distribuyen en las etapas que están descritas en la siguiente tabla:

Fase	Duración
Diseño y planificación	3 semanas
Desarrollo del backend	5 semanas

Desarrollo del frontend	6 semanas
Pruebas y corrección de errores	3 semanas
Implementación y despliegue	3 semanas

Costo

A continuación, se presentan los costos estimados del proyecto:

Concepto	Justificación	Cantidad	Costo Unitario (COP)	Total (COP)
Desarrollador Junior	Su presencia y aportes son fundamentales para llevar a cabo tareas de programación básicas y de mediana complejidad, lo cuál permite que el equipo avance en tareas más rutinarias o repetitivas, dejando a desarrolladores más experimentados como el desarrollador senior poder concentrarse en aspectos críticos del desarrollo. Además, ofrecen un buen balance entre costo y productividad, contribuyendo al desarrollo del proyecto sin inflar demasiado el presupuesto.	1	3’500.000 / mes Precio tomado como referencia del siguiente enlace: https://www.glassdoor.com.ar/Sueldos/colombia-junior-web-developer-sueldo-SRCH_IL.0,8_IN54_KO9,29.htm.	14’000.000
Desarrollador Senior	Un desarrollador senior aporta experiencia y habilidades avanzadas que son necesarias para tomar decisiones técnicas importantes, resolver	1	4’500.000 / mes Precio tomado como referencia del siguiente enlace: https://co.comp	18’000.000

	problemas complejos y supervisar el trabajo de desarrolladores menos experimentados. Su conocimiento profundo en temáticas como la arquitectura de software, patrones de diseño y la puesta en marcha de mejores prácticas de programación garantiza la calidad y la eficiencia del desarrollo.		utrabajo.com/salarios/desarrollador-senior.	
UX/UI Designer	Este integrante del equipo es crucial para crear una interfaz de usuario intuitiva en la aplicación, lo que conlleva a una experiencia de usuario agradable. Su trabajo garantiza que la aplicación no solo sea funcional, sino también fácil de usar y atractiva visualmente, lo cuál es fundamental para retener usuarios y asegurar que la aplicación sea adoptada efectivamente.	1	3’500.000 / mes Precio tomado como referencia del siguiente enlace: https://co.talent.com/salary?job=ux+ui.	14’000.000
Infraestructura (Servidor y BD)	Esta parte es esencial para tener en donde soportar el funcionamiento de la aplicación, lo que incluye el alojamiento del servidor y el almacenamiento seguro de datos. Esto a su vez garantiza la disponibilidad, escalabilidad y seguridad de la aplicación, permitiendo que múltiples usuarios puedan manejar	-	2’000.000 / mes Precio tomado como referencia del siguiente enlace: https://cloud.google.com/pricing.	8’000.000

	grandes volúmenes de datos sin interrupciones.			
Licencias y APIs	Estos componentes son necesarios para poder integrar servicios externos y funcionalidades tanto básicas como avanzadas en la aplicación. Estas herramientas ayudan a mejorar las capacidades de la aplicación al mejorar sus capacidades sin tener que desarrollar todo desde cero.	-	4’000.000 (único) Precio tomado como referencia del siguiente enlace: https://firebase.google.com/pricing?hl=es-419.	4’000.000
Total		-	-	**58’000.000 COP**

Alcance

Se espera que el MVP (Minimum Viable Product) incluya las siguientes funcionalidades:

- Registro y autenticación de usuarios.
- Creación y gestión de perfiles de usuarios.
- Gestión de horarios.
- Búsqueda y consulta de chazas disponibles.
- Sistema de búsqueda y filtrado de candidatos.
- Calificación y reseñas entre usuarios.

En consecuencia, las funcionalidades que quedarían fuera del MVP son las siguientes:

- Sistema de mensajería integrada entre estudiantes postulantes y chazeros.
- Geolocalización de las chazas en un mapa interactivo.
- Historial de postulaciones.
- Sistema de búsqueda y filtrado de productos/servicios.
- Sistema de pedidos en línea.
- Notificaciones sobre disponibilidad y promociones.

5. Casos de Uso del Sistema

Esta actividad ya fue desarrollada según lo indicado, y la información se encuentra en las siguientes rutas:

Documentación de todos los casos de uso desarrollados para el proyecto: https://github.com/javiierbarco/Ingesoft-I/tree/main/Documentaci%C3%B3n/Casos_de_Uso.

Diagrama de casos de uso realizado para el proyecto: https://github.com/javiierbarco/Ingesoft-I/blob/main/Documentaci%C3%B3n/Diagramas/Diagrama_Casos_de_Uso.pdf.

6. Historias de Usuario

El desarrollo de esta actividad se encuentra en la carpeta de Historias de Usuario del repositorio creado para este proyecto, que se encuentra en el siguiente enlace: https://github.com/javiierbarco/Ingesoft-I/tree/main/Documentaci%C3%B3n/Historias_de_Usuario.

7. Clean Code:

El desarrollo de esta actividad se encuentra en el siguiente enlace: https://github.com/javiierbarco/Ingesoft-I/blob/main/Documentaci%C3%B3n/Clean_Code/Informe_codigo_limpio.pdf.

8. Diseño y Arquitectura:

Arquitectura del sistema:

Para la realización de este proyecto se implementó la arquitectura cliente-servidor utilizando. Este enfoque fue elegido por nuestro grupo debido a la necesidad de distribuir las actividades que se van a realizar entre la interfaz de usuario, la lógica de negocio y la gestión de la base de datos, lo que ayuda a facilitar la escalabilidad y el mantenimiento.

Diagrama de arquitectura del sistema:

En el diagrama de la arquitectura del sistema para este proyecto aparecen los siguientes componentes:

Frontend: El frontend es la aplicación móvil desarrollada con Flutter (Dart), que se encarga de UI/UX con Material Design ,manejo de estado con Provider o Riverpod,autenticación con Firebase Auth, consumo de Cloud Firestore para gestionar los datos,notificaciones push con Firebase Cloud Messaging (FCM).

Backend: Consiste principalmente del uso de Firebase como servicio serverless que interactúa con la base de datos que este mismo aporta.

Base de datos: Componente de tipo no relacional que fue desarrollado debido a la necesidad de gestionar turnos en puestos de venta, necesitamos una estructura de datos dinámica donde los usuarios (dueños y trabajadores), los puestos y los turnos puedan organizarse de manera escalable y eficiente.

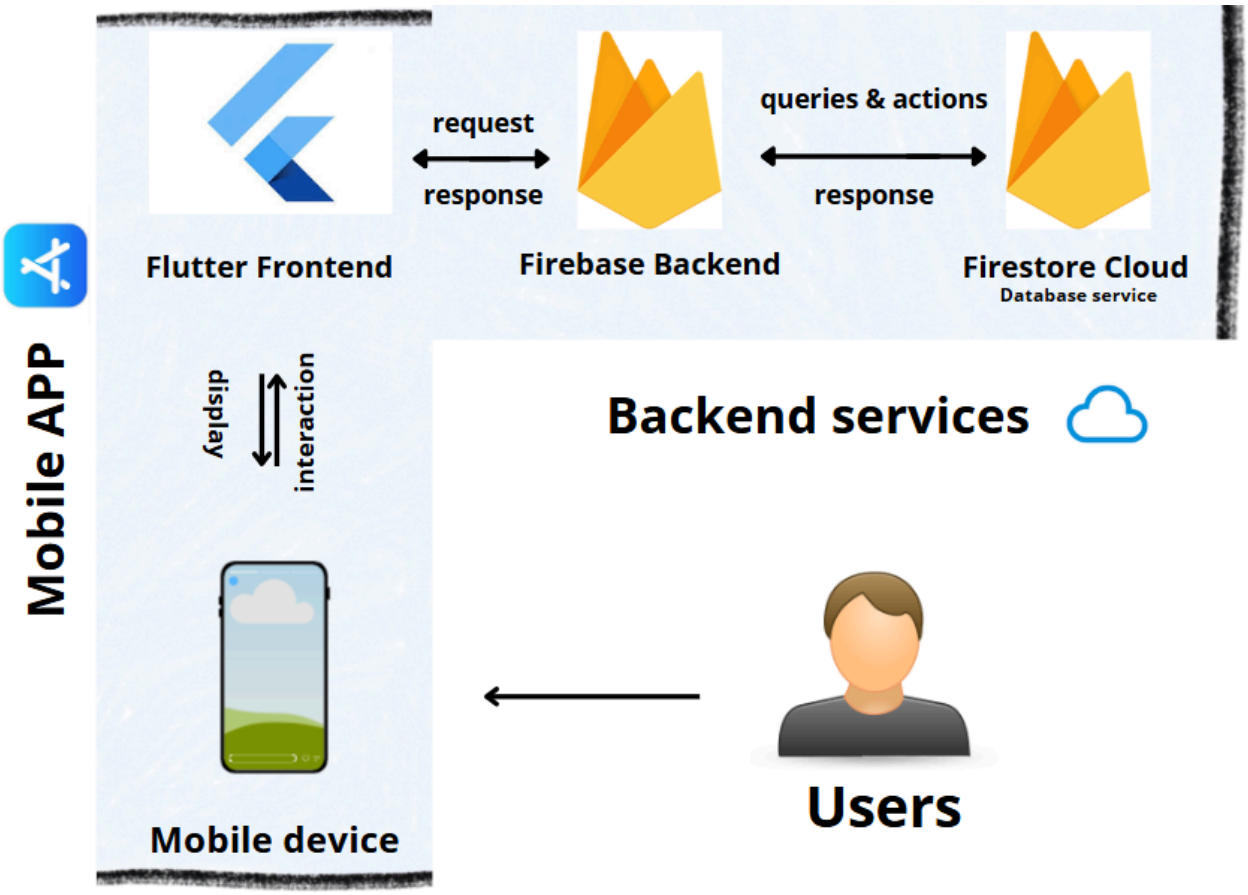

Imagen 3. Diagrama de la arquitectura Cliente-Servidor para el proyecto ChazaUNApp.

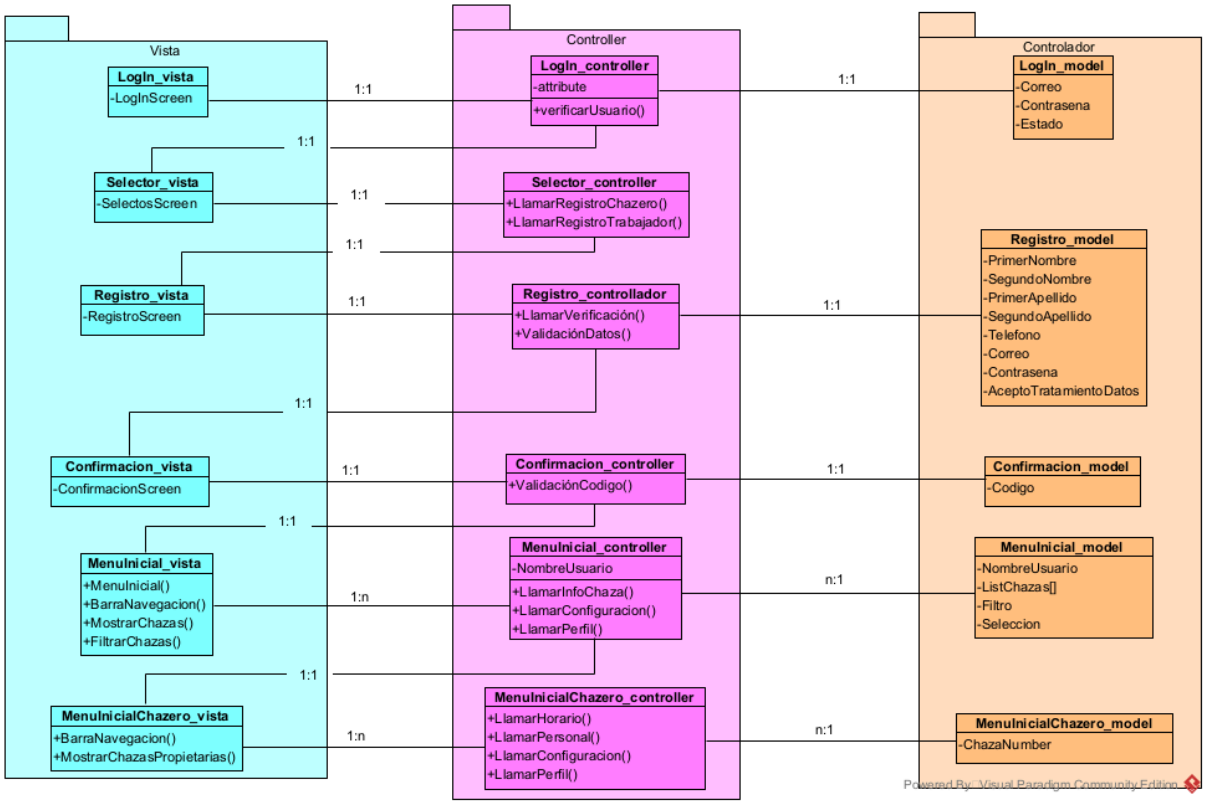

Imagen 4. Diagrama de la arquitectura MVC para el backend del proyecto ChazaUNApp.

Justificación de la arquitectura:

La elección en grupo de una arquitectura Cliente-Servidor se tomó ya que entre todos los integrantes del equipo consideramos que dicha arquitectura facilita el mantenimiento y la escalabilidad del proyecto dado que permite separar la lógica del frontend y del backend. Además, como complemento escogimos la arquitectura MVC porque ayuda a organizar el código y permite modificar cada capa de forma independiente.

Diseño de base de datos:

Para este proyecto, Firestore resulta ideal debido a su flexibilidad y capacidad para manejar datos en tiempo real. Al tratarse de una aplicación que gestiona turnos en puestos de venta, necesitamos una estructura de datos dinámica donde los usuarios (dueños y trabajadores), los puestos y los turnos puedan organizarse de manera escalable y eficiente.

Modelo de base de datos:

El siguiente diagrama no relacional incluye las entidades principales como Usuario, Producto, Orden, y sus relaciones.

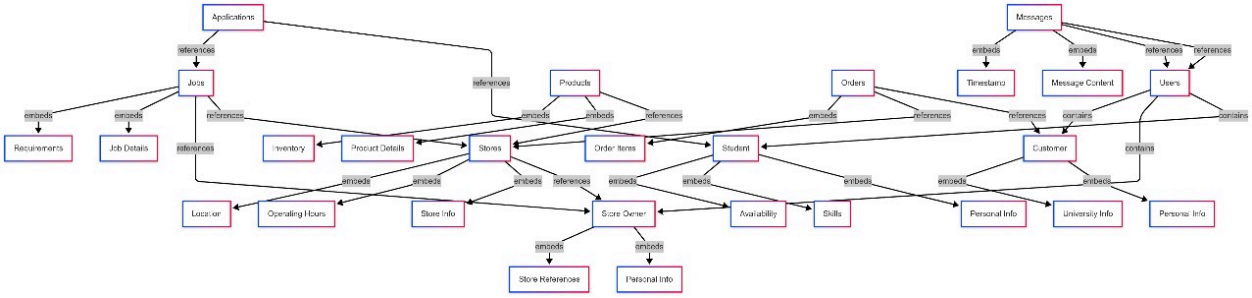

Imagen 5. Diagrama No Relacional para el proyecto ChazaUNApp.

Justificación de la base de datos:

Firestore, al ser una base de datos NoSQL, permite almacenar y consultar documentos sin las restricciones de un esquema fijo, facilitando la organización de turnos dentro de cada puesto sin depender de relaciones complejas. Además, su integración con Firebase Authentication y Cloud Functions nos proporciona un ecosistema seguro y funcional sin necesidad de infraestructura adicional.

9. Patrones de Diseño:

Para este proyecto se han aplicado hasta el momento los siguientes patrones de diseño:

- ***Singleton:*** Se hizo uso del patrón de creación Singleton para la gestión de la conexión a la base de datos, buscando con ello garantizar que solo haya una instancia definida de la conexión a la base de datos en toda la aplicación, evitando con ello la creación de múltiples conexiones que resultan innecesarias.

¿Por qué fue necesario en el proyecto?

Se utiliza el patrón Singleton en la gestión de autenticación **(GAuthService)** y en los servicios de sesión **(services_login.dart, services_menu_inicial.dart, etc.)**. Esto se hace para:

- Evitar la creación de múltiples instancias de los servicios de autenticación y gestión de usuarios.
- Mantener el estado de la sesión del usuario en toda la aplicación.
- Mejorar la eficiencia evitando cargas innecesarias de datos en Firebase.

¿Cómo se implementó?

```
class GAuthService {
  static final GAuthService _instance = GAuthService._internal();
```

```
  factory GAuthService() {
    return _instance;
  }

  GAuthService._internal();
}
```

- Se define una instancia estática **_instance** dentro de la clase.
- El constructor factory factory **GAuthService()** devuelve siempre la misma instancia **(_instance**).
- El constructor privado **GAuthService._internal();** evita que la clase sea instanciada externamente.

Imagen 6. Diagrama UML donde se muestra el uso del patrón Singleton para el proyecto ChazaUNApp.

- ***Observer:*** También se decidió implementar el patrón de comportamiento Observer con el fin de gestionar las notificaciones en tiempo real entre los usuarios, lo que permite que los cambios en la base de datos sean reflejados inmediatamente en la interfaz de usuario sin necesidad de tener que recargar constantemente la aplicación.

¿Por qué fue necesario en el proyecto?

Se usa en la vista **inicio.dart** para:

- Actualizar la UI automáticamente cuando el estado de autenticación del usuario cambia.
- Reducir el acoplamiento entre la lógica de autenticación y la UI.
- Facilitar la gestión del estado en Flutter con **ChangeNotifier** y **notifyListeners()**.

¿Cómo se implementó?

```
import 'package:flutter/material.dart';

class InicioViewModel extends ChangeNotifier {
  String _mensaje = "Bienvenido";
```

```
  String get mensaje => _mensaje;

  void actualizarMensaje(String nuevoMensaje) {
    _mensaje = nuevoMensaje;
    notifyListeners(); // Notifica a los observadores (vistas) que el estado ha cambiado
  }
}
```

- **ChangeNotifier** permite que la clase **InicioViewModel** notifique cambios.
- **notifyListeners()** informa a las vistas que deben redibujar la UI con los nuevos datos.

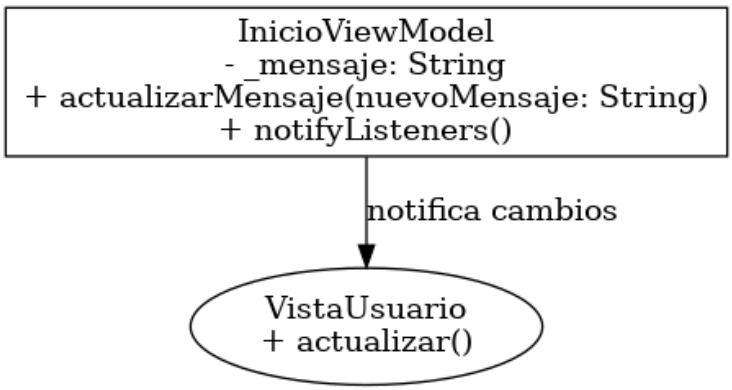

Imagen 7. Diagrama UML donde se muestra el uso del patrón Factory para el proyecto ChazaUNApp.

Diagrama UML:

El siguiente es el diagrama UML de la aplicación que se tiene hasta el momento:

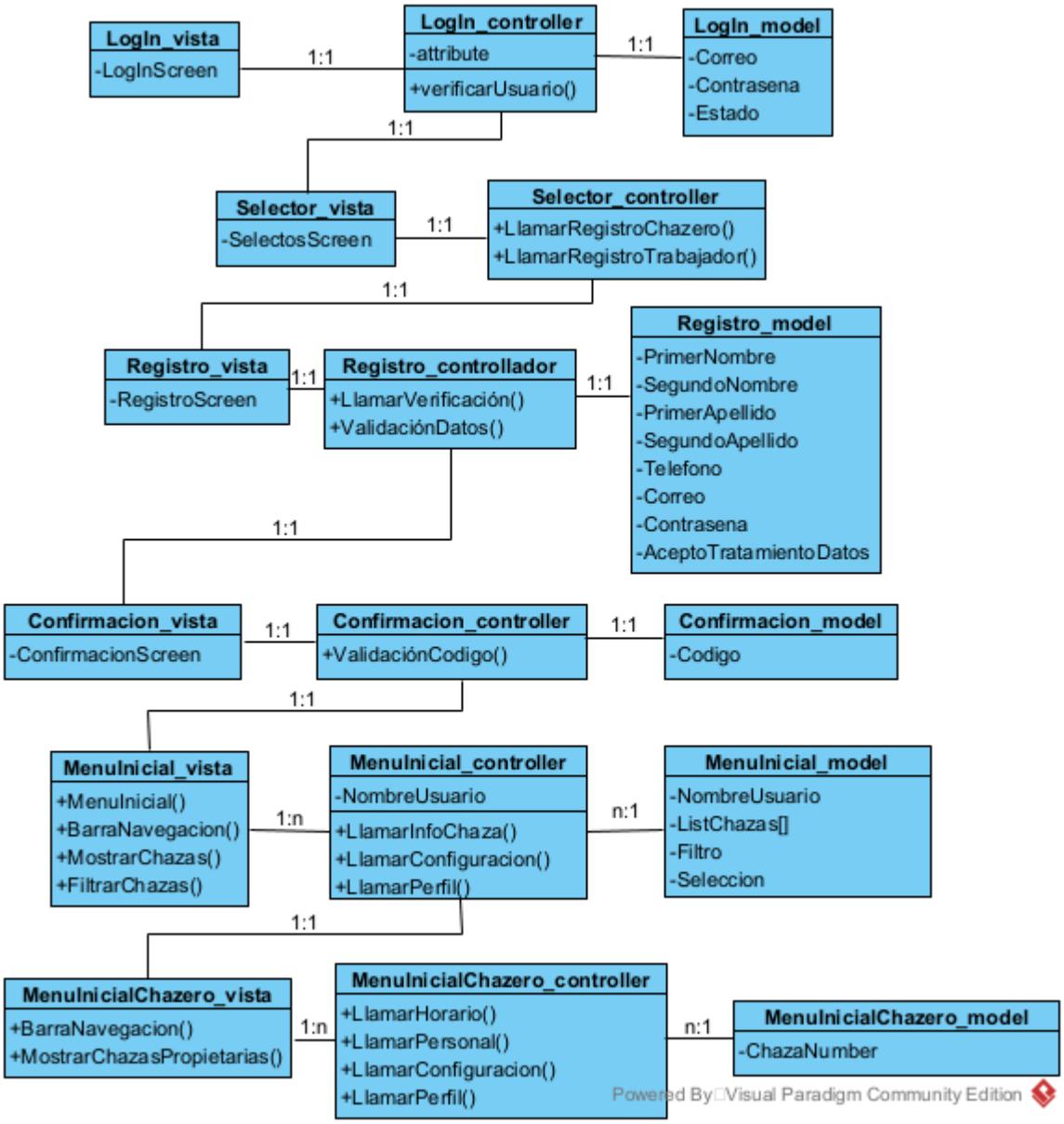

Imagen 8. Diagrama UML de clases para el proyecto ChazaUNApp.