Documentación principal y completa del proyecto

Información General del Proyecto

Nombre: Marcaciones GPP
Descripción: Aplicación Flutter para gestión de marcaciones, asistencia y control de personal
Tipo: Aplicación móvil multiplataforma (Android, iOS, Windows, macOS, Linux, Web)
Versión: 1.0.0+1
Arquitectura: BLoC Pattern
Estado: En desarrollo

Estructura General del Proyecto

```
marcacion_gpp/
├── lib/
│   ├── main.dart                    # Punto de entrada de la aplicación
│   └── src/
│       ├── Desing/                  # Componentes de diseño reutilizables
│       ├── Enums/                   # Enumeraciones
│       ├── Mixins/                  # Mixins para funcionalidades compartidas
│       ├── Models/                  # Modelos de datos generales
│       ├── Pages/                   # Módulos principales (Login, Home, etc)
│       ├── Routes/                  # Sistema de rutas
│       ├── Services/                # Servicios generales
│       ├── SharedPreferences/       # Gestión de preferencias locales
│       ├── Styles/                  # Estilos globales
│       ├── Utils/                   # Utilidades y helpers
│       └── Widgets/                 # Componentes reutilizables
├── assets/                          # Recursos (imágenes, iconos)
├── android/                         # Configuración Android
├── ios/                             # Configuración iOS
├── web/                             # Configuración Web
├── windows/                         # Configuración Windows
├── macos/                           # Configuración macOS
├── linux/                           # Configuración Linux
```

```
└── pubspec.yaml                # Dependencias del proyecto
```

Documentación detallada de cada módulo:

MÓDULO 1: LOGIN

Información General

Nombre del módulo: Login
Descripción: Módulo de autenticación y acceso a la aplicación.
Punto de entrada: Pantalla inicial de la aplicación
Estado: Disponible

Rutas

- Ruta principal: /LoginView
- Nombre constante: LoginView.name
- Tipo: Widget con estado (StatefulWidget)

Ubicación en el Sistema

Menú: Pantalla inicial (sin menú)
Opción: N/A
Acceso: Automático al iniciar la aplicación

Estructura de Carpetas y Archivos - Frontend

lib/src/Pages/Login/

├── LoginNew.dart # Pantalla principal de login

├── Bloc/

│ └── login_bloc.dart # Lógica de negocio (BLoC)

├── Models/

│ ├── domains.dart # Modelo de dominios

```
|   └── station.dart                # Modelo de estación
└── Service/
    └── login_service.dart          # Servicio de comunicación con API
```

Backend - Endpoints API

Base URL: https://sistemacargonew.gppcargo.com/

Funcionalidad	Método	Endpoint	Parámetros
Autenticación	POST	api/login-app-markings	user, password, domain
Obtener Dominios	GET	api/domains	-
Guardar Registro	POST	api/login-app-markings	user, version
Validar Token	POST	api/validate-token	token

Modelos de Datos

Domains.dart

- token: String

- domain: String

- id: int

Station.dart

- id: int

- name: String

- latitude: double

- longitude: double

Funciones Principales

Función	Descripción	Parámetros
loginS()	Autentica el usuario en el sistema	Map<String, dynamic> body
saveRegister()	Guarda el registro de versión	String? user
obtainDomain()	Obtiene los dominios disponibles	-
obtainDomain() (BLoC)	Solicita dominios en el BLoC	BuildContext context

Tablas Relacionadas (Backend)

- login_app_markings: Tabla de autenticación
- domains: Tabla de dominios/clientes
- stations: Tabla de estaciones

Flujo Principal

1. Usuario abre la aplicación
2. Se cargan los dominios disponibles
3. Usuario ingresa credenciales
4. Sistema valida contra API
5. Se guarda sesión en SharedPreferences
6. Se redirige a HomeView

Preferencias Locales Almacenadas

- username: Usuario guardado
- password: Contraseña guardada
- stateUserSaveLogin: Booleano para recordar credenciales
- ultimaPagina: Última página visitada

MÓDULO 2: HOME

Información General

Nombre del módulo: Home / Inicio
Descripción: Panel principal con resumen de marcaciones, información del usuario y navegación a

otros módulos.
Estado: Disponible

Rutas

- Ruta principal: /HomeView
- Nombre constante: HomeView.name
- Tipo: Widget con estado (StatefulWidget)

Ubicación en el Sistema

Menú: Menú desplegable (Drawer)
Opciones:

- Mi Perfil
- Marcaciones
- Asistencia
- Términos y Condiciones
- Cerrar Sesión

Estructura de Carpetas y Archivos - Frontend

lib/src/Pages/Home/

```
├── home_view.dart                # Pantalla principal
├── Bloc/
│   └── home_bloc.dart            # Lógica de negocio
├── Models/
│   └── resumen.dart              # Modelo de resumen de marcaciones
├── Services/
│   └── home_services.dart        # Servicios de API
└── Widgets/
    ├── boxes.dart                # Cajas de información
    ├── drawer.dart               # Menú desplegable
    ├── newPost.dart              # Componente de nuevas publicaciones
```

```
├── post_target.dart              # Componente de publicaciones dirigidas
└── tile.dart                     # Componentes tipo lista
```

Backend - Servicios

El módulo Home utiliza servicios de otros módulos:

- LoginService: Para validar sesión
- PerfilService: Para obtener datos del usuario
- AsistenciaService: Para estadísticas de asistencia

Modelos de Datos

AttendanceResume (resumen.dart)

- totalMarcaciones: int

- marcacionesHoy: int

- estadoActual: String

- horaUltimaActividad: DateTime

Person (person.dart)

- id: int

- userid: String

- nombre: String

- apellido: String

- email: String

- telefono: String

Funciones Principales

Función	Descripción
obtenerResumen()	Obtiene el resumen de marcaciones del usuario
cargarDatosUsuario()	Carga información del perfil
cargarEstadisticas()	Obtiene estadísticas de asistencia
cerrarSesion()	Cierra la sesión del usuario

Componentes Visuales

- Header: Información del usuario y bienvenida
- Cajas de Estadísticas: Resumen de marcaciones
- Drawer: Menú de navegación lateral
- Confetti Animation: Animación de celebración

Tablas Relacionadas (Backend)

- users: Información de usuarios
- markings: Historial de marcaciones
- attendance: Asistencia del usuario

MÓDULO 3: ASISTENCIA

Información General

Nombre del módulo: Asistencia
Descripción: Módulo para visualizar y gestionar asistencia con búsqueda por fecha y calendario interactivo.
Estado: Disponible

Rutas

- Ruta principal: /AsistenciaView
- Nombre constante: AsistenciaView.name
- Tipo: Widget con estado (StatefulWidget)

Ubicación en el Sistema

Menú: Accesible desde Home → Drawer → Asistencia
Ubicación: Tercera opción en el menú principal

Estructura de Carpetas y Archivos - Frontend

lib/src/Pages/Asistencia/

├── asistencia_view.dart # Pantalla principal

├── Bloc/

│ └── asistencia_bloc.dart # Lógica de negocio

├── Models/

│ └── search_model.dart # Modelo de búsqueda

├── Services/

│ └── asistencia_service.dart # Servicios de API

└── Widgets/

├── box_asistant.dart # Cajas de asistencia

├── calendart.dart # Calendario interactivo

└── openCamera.dart # Acceso a cámara

Backend - Endpoints API

Base URL: https://sistemacargonew.gppcargo.com/

Funcionalidad	Método	Endpoint	Parámetros
Buscar Asistencia	POST	api/search-markings-attendance-by-user-date	date, user, token

Modelos de Datos

SearchModel (search_model.dart)

- date: String

- markings: List<Marking>

  - type: String (Entrada, Salida)

  - time: DateTime

  - status: String

  - location: String

Funciones Principales

Función	Descripción	Parámetros
searcAsistence()	Busca asistencias por fecha	String date
selectDate()	Selecciona una fecha del calendario	DateTime date
filterByRange()	Filtra por rango de fechas	DateTime start, DateTime end

Características

- Calendario interactivo: Selección visual de fechas
- Búsqueda por fecha: Obtiene marcaciones de un día específico
- Visualización de historial: Lista detallada de entradas/salidas
- Indicadores de estado: Puntual, Tarde, Ausente

Tablas Relacionadas (Backend)

- markings: Registros de marcaciones
- attendance: Registro de asistencia
- users: Información del usuario

MÓDULO 4: MARCACIONES

Información General

Nombre del módulo: Marcaciones
Descripción: Módulo para realizar marcaciones de entrada/salida con opciones de QR, geolocalización, cámara y registro de motivos.
Estado: Disponible
Permisos requeridos: Cámara, GPS, Almacenamiento

Rutas

- Ruta principal: /MarcacionesGPP
- Nombre constante: MarcacionesGPP.name
- Tipo: Widget con estado (StatefulWidget)

Ubicación en el Sistema

Menú: Accesible desde Home → Drawer → Marcaciones
Ubicación: Segunda opción en el menú principal

Estructura de Carpetas y Archivos - Frontend

lib/src/Pages/Marcaciones/

├── marcaciones_view_GPP.dart # Pantalla principal (511 líneas)

├── Bloc/

│ └── marcaciones_Bloc.dart # Lógica de negocio

├── Logic/

│ └── marcaciones_Logic.dart # Lógica de procesamiento

├── Models/

│ ├── marcaciones_Model.dart # Modelo de marcación

│ └── marcaciones_obtain.dart # Modelo de datos obtenidos

├── Services/

│ └── Marcaciones_Services.dart # Servicios de API (120 líneas)

└── Widgets/

└── (Widgets específicos del módulo)

Backend - Endpoints API

Base URL: https://sistemacargonew.gppcargo.com/
Base URL (RRHH): https://rrhh.gppcargo.com/

Funcionalidad	Método	Endpoint	Parámetros
Crear Marcación	POST	api/marking	type_marking, user, station, file_id, token, motive
Buscar Marcaciones	POST	api/search-markings-attendance-by-user	date, user, token
Obtener Datos Actuales	POST	api/markings-date-now-by-user	user, token
Validar Geovalla	POST	api/verify-latitude-longitude-geofence	latitude, longitude, station, token
Validar QR	POST	api/method/rr_hh.recursos_humanos.doctype.asistencia.api.validate_qr_by_station	qr_id, station_id, fecha
Validar Uso Fuera de Sede	POST	api/method/rr_hh.recursos_humanos.doctype.asistencia.api.validate_use_number	number, token

Modelos de Datos

MarcacionesObtain

- id: int

- user: String

- type: String (Entrada, Salida)

- timestamp: DateTime

- latitude: double

- longitude: double

- station: int

- motive: String

- status: String

MarcacionesModel

- typeMarking: String

- userId: String

- stationId: int

- fileId: String

- motive: String

- token: String

Funciones Principales

Función	Descripción	Parámetros
generateMarcacion()	Genera una nueva marcación	String type, String file, String motive
validateGeocerca()	Valida si está dentro de la geovalla	num latitude, num longitude
validateQR()	Valida un código QR	String scanQR, String fecha
validateToken()	Verifica validez del token	-
showMarcaciones()	Obtiene marcaciones por fecha	String fecha
obtainDataMarcaciones()	Obtiene marcaciones actuales	-
validateFueraSede()	Valida si puede usar función fuera de sede	-

Opciones de Marcación

1. Por QR: Escaneo de código QR
2. Por Geolocalización: Validación de ubicación GPS
3. Por Cámara: Captura de foto como evidencia
4. Con Motivo: Registro de razón de marcación

Validaciones

- Verificación de permisos de cámara
- Verificación de permisos de GPS
- Validación de token
- Validación de geovalla (geofence)
- Validación de QR por estación
- Validación de uso fuera de sede

Tablas Relacionadas (Backend)

- markings: Registros de marcaciones
- users: Información de usuarios
- stations: Estaciones de trabajo
- qr_codes: Códigos QR registrados
- geofence: Áreas permitidas
- marking_motives: Motivos de marcación

MÓDULO 5: PERFIL

Información General

Nombre del módulo: Perfil / Mi Cuenta
Descripción: Módulo para visualizar y editar información personal del usuario.
Estado: Disponible

Rutas

- Ruta principal: /PerfilGPP
- Nombre constante: PerfilGPP.name
- Tipo: Widget con estado (StatefulWidget)

Ubicación en el Sistema

Menú: Accesible desde Home → Drawer → Mi Perfil
Ubicación: Primera opción en el menú principal

Estructura de Carpetas y Archivos - Frontend

```
lib/src/Pages/Perfil/
├── perfilnewGpp.dart                # Pantalla principal
├── Bloc/
│   └── perfil_bloc.dart             # Lógica de negocio
├── Models/
│   ├── perfil.dart                  # Modelo de datos del perfil
│   └── person.dart                  # Modelo de persona
├── Services/
│   └── perfil_service.dart          # Servicios de API
└── Widgets/
    ├── header.dart                  # Encabezado con foto
    └── list_deteails.dart           # Lista de detalles
```

Backend - Servicios

Los servicios de perfil se integran con:

- LoginService: Para validación de credenciales
- Endpoints de usuario en api/user o similar

Modelos de Datos

PerfilDatos (perfil.dart)

- id: int

- nombre: String

- apellido: String

- email: String

- telefono: String

- departamento: String

- puesto: String

- fechaIngreso: DateTime

- fotoUrl: String

- estado: String

Person (person.dart)

- id: int

- userid: String

- nombre: String

- apellido: String

- email: String

- telefono: String

- departamento: String

- rol: String

Funciones Principales

Función	Descripción
obtenerDatosUsuario()	Obtiene información completa del usuario
actualizarPerfil()	Actualiza los datos del perfil
cambiarContraseña()	Cambia la contraseña del usuario
cargarFoto()	Carga la foto de perfil

Información Mostrada

- Foto de perfil

- Nombre completo
- Email
- Teléfono
- Departamento
- Puesto
- Fecha de ingreso
- Estado del usuario

Tablas Relacionadas (Backend)

- users: Información de usuarios
- employees: Datos de empleados
- departments: Departamentos

MÓDULO 6: TÉRMINOS Y CONDICIONES

Información General

Nombre del módulo: Términos y Condiciones
Descripción: Módulo para visualizar y aceptar términos y condiciones de uso.
Estado: Disponible

Rutas

- Ruta principal: /TerminosView (inferido)
- Nombre constante: TerminosView.name
- Tipo: Widget con estado (StatefulWidget)

Ubicación en el Sistema

Menú: Accesible desde Home → Drawer → Términos y Condiciones
Ubicación: Última opción en el menú principal

Estructura de Carpetas y Archivos - Frontend

lib/src/Pages/TerminosYCondiciones/

├── termino_View.dart # Pantalla principal

├── Bloc/

│ └── (Lógica de negocio)

└── Services/

└── (Servicios específicos)

Funciones Principales

- Visualizar términos y condiciones
- Aceptar términos
- Rechazar y cerrar sesión

Información Asociada

- Versión de términos
- Fecha de última actualización
- Estado de aceptación del usuario

Servicios y Utilidades Globales

Services Generales

Ubicación: lib/src/Services/general_Services.dart

- Servicios genéricos reutilizables
- Manejo de conexiones
- Procesamiento de datos globales

Utilidades

Ubicación: lib/src/Utils/

```
├── check_connection.dart          # Verificar conexión a internet
├── column_builder.dart            # Constructor de columnas dinámicas
├── downloader_Image.dart          # Descarga de imágenes
├── icon_string.dart               # Conversión de strings a iconos
├── imagenes_string.dart           # Manejo de imágenes
├── ineternet.dart                 # Utilidades de red/HTTP
├── openCamera.dart                # Acceso a cámara
├── permission.dart                # Gestión de permisos
├── qr_scan.dart                   # Escaneo de QR
├── showpdf.dart                   # Visualización de PDF
├── urls_servers.dart              # URLs de servidores
├── urls.dart                      # URLs de aplicación
└── ErrorHandler/                  # Manejo centralizado de errores
    ├── bloc/
    │   └── error_handler_bloc.dart
```

```
├── Models/
│   └── error_handler_model.dart
└── Services/
```

SharedPreferences

Ubicación: lib/src/SharedPreferences/preferences.dart

Almacenamiento local de datos:

- username: Usuario actual
- password: Contraseña (si está habilitado)
- token: Token de sesión
- userId: ID del usuario
- stateUserSaveLogin: Recordar credenciales
- ultimaPagina: Última página visitada

Widgets Reutilizables

Ubicación: lib/src/Widgets/

```
├── boton_general.dart          # Botón estándar
├── Json.dart                   # Procesamiento de JSON
├── layoutView.dart             # Layout principal con drawer
├── text_input.dart             # Campo de entrada de texto
├── Files/
│   ├── file_Manager.dart       # Gestor de archivos
│   ├── gallery.dart            # Galería de imágenes
│   ├── media_buttom.dart       # Botones multimedia
│   └── recorder_page.dart      # Grabador de audio/video
├── Imputs/
│   └── outline_input.dart      # Input con outline
```

```
├── Layouts/
│   └── showpdf.dart              # Layout para PDF
├── Pdfs/
│   └── visor_pdfs.dart           # Visor de PDF
└── Temp/
    └── overskull.dart            # Componentes temporales
```

Estilos y Diseño

Ubicación: lib/src/Styles/styles.dart y lib/src/Desing/

```
├── desing_text.dart              # Estilos de texto
├── flush.dart                    # Colores y estilos generales
└── speech.dart                   # Síntesis de voz
```

Arquitectura de Comunicación

Patrón BLoC (Business Logic Component)

Todos los módulos implementan el patrón BLoC:

View (UI) <--> BLoC <--> Service <--> API/Backend

|

v

SharedPreferences (Almacenamiento local)

Flujo de Datos

1. UI (View) solicita datos
2. BLoC recibe la solicitud y coordina la lógica
3. Service realiza la llamada HTTP
4. Respuesta es procesada por el BLoC
5. UI se actualiza con los nuevos datos

Método HTTP Personalizado

Ubicación: lib/src/Utils/ineternet.dart

Internet.httpPost(

url: String,

body: Map<String, dynamic>,

timeOut: bool (opcional),

seconds: int (opcional)

)

```
Internet.httpGet(
  url: String,
  body: String
)
```

Dependencias Principales

camera: ^0.10.5 # Acceso a cámara

dio: 5.6.0 # Cliente HTTP

flutter_pdfview: ^1.2.1 # Visualización de PDF

geolocator: ^9.0.2 # Geolocalización

image_picker: 1.0.0 # Selección de imágenes

permission_handler: ^9.2.0 # Gestión de permisos

qr_code_scanner: ^1.0.1 # Escaneo de QR

shared_preferences: ^2.0.18 # Almacenamiento local

table_calendar: ^3.0.8 # Calendario interactivo

device_preview: ^1.1.0 # Preview de dispositivos

confetti: ^0.6.0 # Animaciones de confeti

Seguridad

Token de Sesión

- Se almacena en SharedPreferences
- Se incluye en cada solicitud API
- Se valida en endpoints específicos

Permisos Requeridos

Android:

- CAMERA: Para captura de fotos
- ACCESS_FINE_LOCATION: Para GPS
- READ_EXTERNAL_STORAGE: Para acceso a archivos

iOS:

- NSCameraUsageDescription: Para cámara
- NSLocationWhenInUseUsageDescription: Para GPS

Manejo de Errores

- Bloque FlutterError.onError: Captura errores de UI
- runZonedGuarded: Captura errores asincronos
- ErrorHandlerBloc: Centraliza el manejo de errores

Configuración por Plataforma

Android

Ubicación: android/app/build.gradle

- Compilación mínima SDK 21
- Target SDK 33+

iOS

Ubicación: ios/Podfile

- Deployment target 11.0+
- Frameworks requeridos: CoreLocation, AVFoundation

Windows/macOS/Linux

Configuraciones en sus respectivos directorios con soporte nativo.

Guía de Instalación y Ejecución

Requisitos Previos

Flutter SDK >= 3.13.6

Dart SDK >= 3.1.3

Android SDK / Xcode

Git

Instalación

```
# Clonar repositorio

git clone <url-repositorio>


# Instalar dependencias

flutter pub get


# Generar archivos necesarios

flutter pub run build_runner build


# Ejecutar aplicación

flutter run
```

Build para Producción

```
# Android

flutter build apk --release

flutter build appbundle --release


# iOS

flutter build ios --release


# Web

flutter build web --release


# Windows

flutter build windows --release
```

Flujos Principales

Flujo de Inicio de Sesión

1. Aplicación abre → LoginView

2. Cargar dominios disponibles

3. Usuario ingresa credenciales

4. Validar contra API

5. Guardar sesión

6. Navegar a HomeView

Flujo de Marcación

1. Usuario abre Marcaciones

2. Sistema solicita permisos (cámara, GPS)

3. Usuario selecciona tipo de marcación:

   a) Por QR → Escanear código

   b) Por Geovalla → Validar ubicación

   c) Por Cámara → Capturar foto

4. Sistema valida opciones seleccionadas

5. Usuario ingresa motivo (opcional)

6. Registrar marcación en API

7. Mostrar confirmación

Flujo de Consulta de Asistencia

1. Usuario abre Asistencia

2. Seleccionar fecha en calendario

3. Sistema consulta asistencias

4. Mostrar historial de marcaciones

5. Opciones de filtrado (rango de fechas)

Manejo de Errores

Tipos de Errores Capturados

1. Errores de UI: FlutterError.onError
2. Errores Asincronos: runZonedGuarded
3. Errores de Red: Validación en Internet.httpPost/Get
4. Errores de Validación: Por servicio

Respuesta a Errores

Estructura estándar de respuesta:

```
{

  "valor": true/false,

  "message": "Descripción del error",

  "data": {}

}
```

Notas Importantes

1. Token de Sesión: Crítico para todas las operaciones
2. Geolocalización: Requiere permisos específicos del usuario
3. Cámara y Galería: Necesarios para evidencia de marcación
4. Conectividad: Aplicación requiere conexión a internet
5. Almacenamiento Local: Usa SharedPreferences para datos críticos