**Documento Técnico**

**Recomendación de Restaurantes**

**App Móvil**

Restaurant

Login

LoginActivity

LoginFragment

RestaurantFragment

RestaurantActivity

RestaurantPresentere

LoginPresenter

Restaurants

Signup

SignupActivity

SignupFragment

SignupPresenter

RestaurantsActivity

RestauranstPresentere

RestaurantsFragment

RestaurantsAdapter



Map

Settings

Userprofile

SettingsActivity

UserProfileActivity

MapActivity

UserProfileFragment

MapFragment

Splash

SplashScreenActivity

UserProfilePresenter

MapPresenter

Data

Data

Source

Model

Restaurant

RestaurantsRemoteSource

RestaurantsLocalSource

RestaurantsRepository

RestaurantsProvider

RestaurantsDbHelper

User

**Catálogo de elementos:**

**Splash:** Contiene la pantalla para mostrar un splashcreen

**Activity:** Representa una pantalla que aparace al abrir la aplicacion con el logo de y otros datos de esta.

**Login:** Contiene los elementos que permite al usuario autenticarse y obtener acceso a los contenidos de la aplicación.

**LoginActivity:** Representa la pantalla principal que contiene los componentes para autenticar al usuario.

**LoginFragment:** Representa la vista que permite al usuario introducir sus datos para autenticarse.

**RestaurantPresenter:** Se encarga de manejar la lógica de autenticación y administar la vista del login.

**Signup:** Contiene los elementos que permite a los nuevos usuarios registrarse e iniciar a usar la aplicación.

**SignupActivity:** Representa la pantalla principal que contiene los componentes para registrar al usuario.

**SignupFragment:** Representa la vista que permite al usuario introducir sus datos para registrarse.

**SignupPresenter:** Se encarga de manejar la lógica de envió de registro y administar la vista del registro.

**Restaurant:** Contiene los elementos para mostrar los detalles de un restaurante y permitir puntuarlo.

**RestaurantActivity:** Representa la pantalla principal que contiene los componentes para mostrar los datos un restaurante.

**RestaurantFragment:** Representa la vista que contiene los datos de un restaurante.

**RestaurantPresenter:** Se encarga de manejar la lógica del negocio y las interacciones entre la vista del restaurante y el modelo de datos.

**Restaurants:** Contiene los elementos para mostrar en forma de lista los restaurantes recomendados.

**RestaurantsActivity:** Representa la pantalla principal que contiene los componentes para mostrar la lista de restaurantes.

**RestaurantsFragment:** Representa la vista que contiene la lista de restaurantes.

**RestaurantAdaptaer:** Se encarga de enlazar una colección de restaurantes y una lista de sus vistas.

**RestaurantPresenter:** Se encarga de manejar la lógica del negocio y las interacciones entre la vista de los restaurantes y la colección de datos.

**Map:** Contiene los elementos para mostrar en el mapa uno o varios restaurantes.

**MapActivity:** Representa la pantalla principal que contiene los componentes para mostrar un mapa.

**MapFragment:** Representa la vista que contiene muestra el mapa.

**MapPresenter:** Se encarga de manejar los datos que se van a presentar en el mapa y las interacciones con la vista.

**UserProfile:** Contiene los elementos para mostrar los datos del usuario y editarlos.

**UserProfileActivity:** Representa la pantalla principal que contiene los componentes para mostrar los datos del usuario autenticado.

**UserProfileFragment:** Representa la vista que muestra los datos del usuario autenticado.

**UserProfilePresenter:** Se encarga de manejar la lógica para mostrar y editar los datos del usuario.

**Data:** Contiene los elementos para manejar el almacenamiento y obtención de datos que serán utilizados para la aplicación de forma estructurada.

**Source:** Contiene los elementos que administran la persistencia de datos en orígenes tanto locales como remotos.

**RestaurantsProvider**: Componente que provee una interfaz para compartir con otras aplicaciones y almacenar datos de forma local en una base de datos.

**RestaurantsDbHelper**: Elemento para manejar la creación y conexión con la base de datos local (sqlite).

**RestaurantsRepository**: Se encarga de proporcionar y persistir los restaurantes y manejar su origen sea local o remoto.

**RestaurantsLocalSource**: Administra la persistencia y obtención de restaurantes en la base de datos local.

**RestaurantsRremoteSource**: Administra la persistencia y obtención de restaurantes en fuentes remotas.

**Model:** Representa la pantalla principal que contiene los componentes que permiten al usuario establecer sus preferencias.

**Restaurant:** Encapsula los datos que representan un restaurante del mundo real.

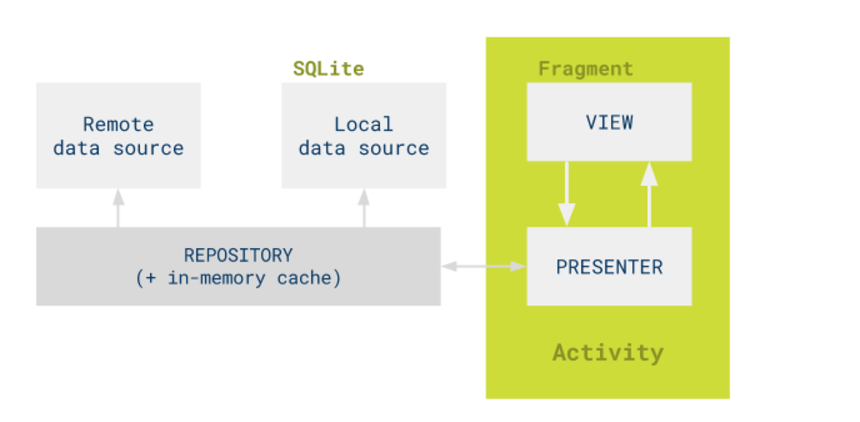
**User:** Encapsula los datos que representan un usuario en el mundo real.

**Settings:** Contiene los elementos para administrar la configuración del usuario.

**SettingsActivity:** Representa la pantalla principal que contiene los componentes que permiten al usuario establecer sus preferencias.

**Arquitectura dinámica**

**La aplicación será desarrollada utilizando el patrón de diseño Modelo Vista Presentador (MVP)**



MVP es un patrón arquitectónico de interfaz de usuario diseñada para facilitar pruebas de unidad automatizada y mejorar la separación de inquietudes en lógica de presentación:

El modelo es una interfaz que define los datos que se mostrará o no actuado en la interfaz de usuario.

El presentador actúa sobre el modelo y la vista. Recupera datos de los repositorios (el modelo), y los formatea para mostrarlos en la vista.

La vista es una interfaz pasiva que exhibe datos (el modelo) y órdenes de usuario de las rutas (eventos) al presentador para actuar sobre los datos.

**Descripción de Tecnologías**

La aplicación será desarrollada con las siguientes tecnologías:

Capa de Presentación

**App:** C# como lenguaje de programación en un proyecto Xamarin.

**Persistencia:** para persistir los datos se utilizara una base de datos sqlite.

**Web App**



Address

RestaurantType

Food Type

Restaurant

UserReview

UserRole

User

Model

RestaurantsRepository

UsersRepository

ContactView

SettingsView

View

LoginView

SignUpView

ProfileView

MapView

RestaurantView

RestaurantsView

Data

MapController

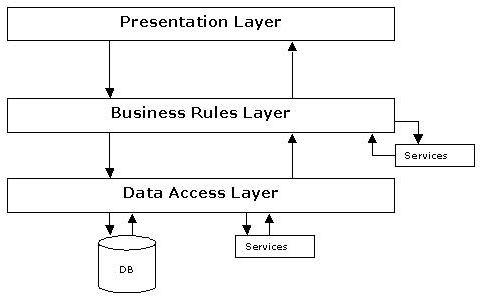
AccountController

RestaurantsController

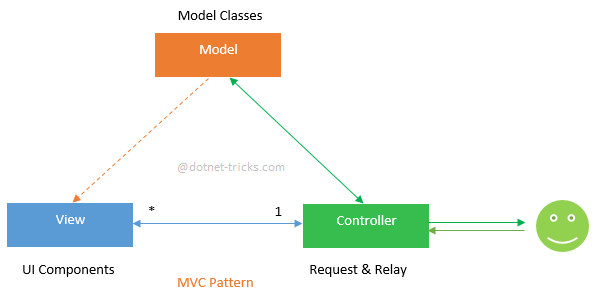
Controller

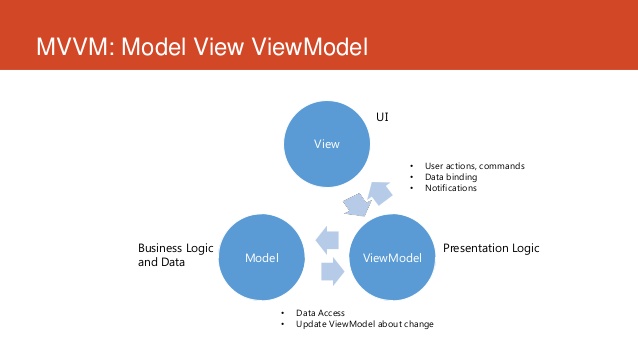
RestaurantApp

La Arquitectura de la aplicación estará estructurada de la manera de N-Capas utilizando 3 capas: Data, Negocios y Presentación.



A su vez la aplicación se manejará utilizando el patrón diseño MVC para el backend y el patron de diseño MVVM para el front-end





**Descripción de Tecnologías**

La aplicación será desarrollada con las siguientes tecnologías:

Capa de Presentación

**Back-End:** C# como lenguaje de programación en ASP.Net MVC5.

**Front-End:** JavaScript como lenguaje de programación , Angularjs como framework mvvm , HTML y CSS.

**Persistencia:** para persistir los datos se utilizara una base de datos estructurada corriendo en SQL SERVER 2012 y se utilizara EntityFramework Como ORM para manejar la interacción con esta.