DOCUMENTACIÓN TÉCNICA

1. Arquitectura

Utiliza el patron **MVVM (Model–View–ViewModel)**.

1.1. Model

La capa **Model** contiene las entidades del dominio y los repositorios de acceso a datos.

Características:

- Representa los datos y las reglas básicas del sistema
- Incluye las entidades principales y los repositorios
- No depende de la interfaz gráfica ni de los ViewModels

Esta capa centraliza la definición de la información utilizada por la aplicación.

1.2. View

La capa **View** define la interfaz gráfica de la aplicación y está compuesta por archivos **XAML**.

Características:

- Muestra la información al usuario
- No contiene lógica de negocio
- Se enlaza a los ViewModels mediante **DataBinding**

Gracias al enlace de datos, la vista se actualiza automáticamente cuando cambian los datos.

1.3. View-Model

La capa **ViewModel** actúa como intermediaria entre la vista y el modelo y contiene la lógica de presentación de la aplicación.

Características:

- Gestiona el estado y el comportamiento de la aplicación
- Se comunica con los modelos y repositorios
- Expone propiedades observables
- Define y gestiona los **comandos (ICommand)** utilizados por la interfaz

Los comandos están integrados dentro de los ViewModels para simplificar la estructura del proyecto.

1.4. Fuera del Patrón MVVM

Además del proyecto principal, la solución incluye otros proyectos que no forman parte directa del patrón MVVM, pero que complementan la aplicación.

1.4.1. Informes

Proyecto encargado de la **generación de informes**.

Características:

- Accede a los **DataSets** para obtener la información
- Genera listados y documentos con los datos del sistema
- Está desacoplado de la interfaz gráfica

1.1.1. Test

Proyecto dedicado a las **pruebas unitarias**.

Características:

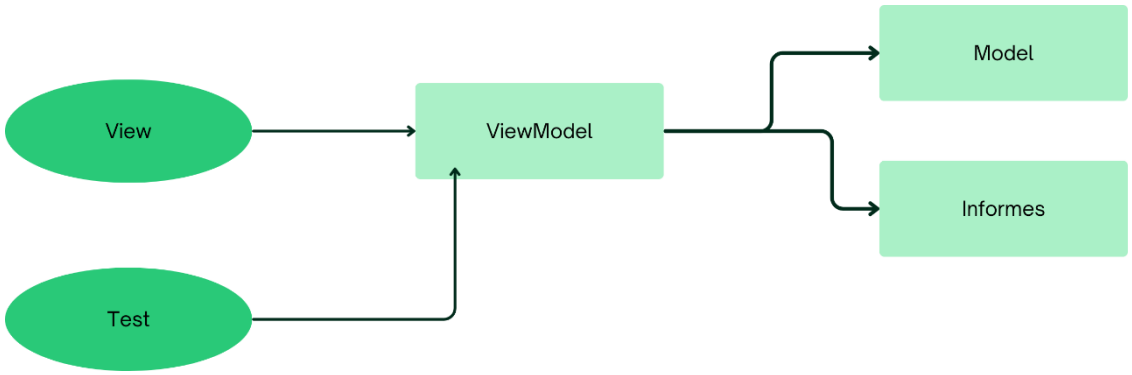
- Contiene pruebas automatizadas
- Verifica el correcto funcionamiento de la lógica y los repositorios
- No incluye interfaz gráfica.

2. Diagrama de capas

View → ViewModel → Model → Base de Datos

View → ViewModel → Informes

Tests → ViewModel



3. Carpetas

- **Models/** — entidades EF, Model1.edmx, ReservaRepository, SocioRepository.
- **ViewModel/** — BaseViewModel, ReservasViewModel, SociosViewModel, RelayCommand.
- **Views/** — XAML, ventanas principales (MainWindow.xaml), user controls.
- **Informes/** — DataSets tipados y repositorios para generación de informes.
- ZenithZone.Test/ — pruebas unitarias; App.config para EF en tests.
- **docs/** o **assets/** — documentación e imágenes (añadir carpeta física).

4. Explicacion Tecnica

4.1. Models

4.1.1. Socios

Tiene Informacion de cada Socio, su nombre, su Email o si esta activo

4.1.2. Actividades

Tiene Informacion de Las Actividades, Nombre de la Actividad y Aforo máximo

4.1.3. Reservas

Relaciona Socios Con Actividades y Fechas en las que el Socio ha reservado para hacer dicha actividad

4.2. ViewModels

Responsabilidades: exponer **ObservableCollection<T>**, **INotifyPropertyChanged**, **ICommand**, validar reglas de negocio.

Ejemplo: ReservasViewModel contiene validaciones (fecha no pasada, unicidad por día, aforo).

4.3. Commands

RelayCommand implementa **ICommand** encapsulando la acción (Execute) y la condición (CanExecute). En los **ViewModel** se exponen comandos como GuardarCommand, EliminarCommand y se enlazan desde la **View** para ejecutar lógica sin code-behind.

4.4. Acceso a Datos

El acceso a datos se realiza con Entity Framework 6 mediante el contexto zenithzoneEntities. Las clases ***Repository** (por ejemplo ReservaRepository, ActividadRepository) encapsulan operaciones CRUD y validaciones que requieren consultas a BD (ValidarReserva, ValidarAforoDisponible). La cadena de conexión vive en App.config del ensamblado ejecutable o del proyecto de tests.

5. Diagrama de Base de Datos.