

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Clase 08: Modelo-Vista-Controlador

Francisco Gazitúa Requena (fco_gazitua_requena@uc.cl)

IIC2113 Diseño Detallado de Software

28 de Agosto, 2025

Modelo-Vista-Controlador

Modelo-Vista-Controlador

Motivación: En cada programa podemos encontrar tres responsabilidades principales.

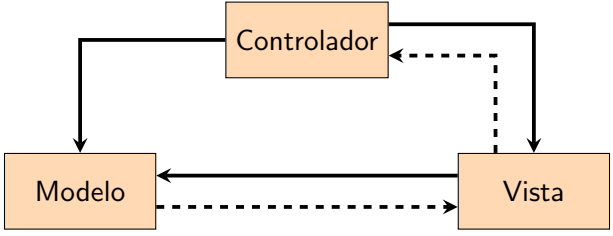
- La **lógica** de la aplicación (qué hacer).
- Los **datos** sobre los que se opera (y cómo se organizan).
- La **presentación** de la aplicación (como se expone al mundo).

Modelo-Vista-Controlador

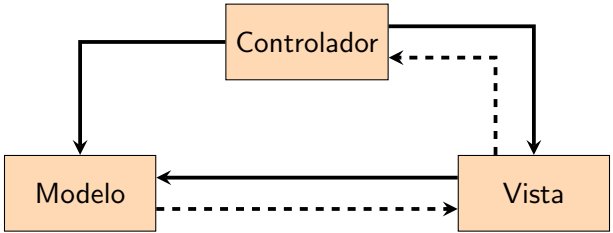
Motivación: En cada programa podemos encontrar tres responsabilidades principales.

- La **lógica** de la aplicación (qué hacer).
- Los **datos** sobre los que se opera (y cómo se organizan).
- La **presentación** de la aplicación (como se expone al mundo).

El patrón MVC separa estas 3 responsabilidades y nos dice cómo debiesen interactuar.



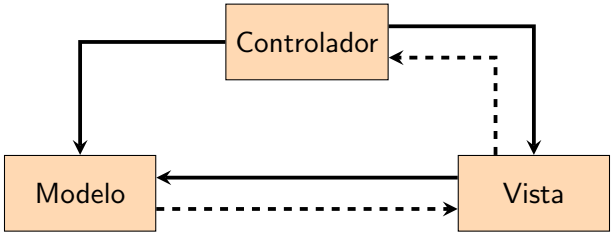
Modelo-Vista-Controlador



Controlador:

- El controlador dirige la orquesta de la aplicación.
- Define qué se hace y cuándo.
- Conoce tanto al modelo como a la vista (o a una interfaz que represente la vista).
- Puede recibir notificaciones de la vista (e.j., se hizo un click).

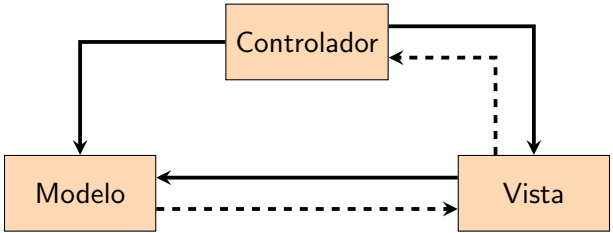
Modelo-Vista-Controlador



Modelo:

- El modelo representa los datos sobre los que se va a operar.
  - Encapsular cualquier forma de almacenamiento de datos.
  - Entregar valor agregado (e.j., hace cálculos sobre los datos).
- El modelo es actualizado/modificado por el controlador.
- El modelo puede notificar a la vista si sus datos cambian.
- El modelo no conoce ni al controlador ni a la vista (evitando acoplamiento).

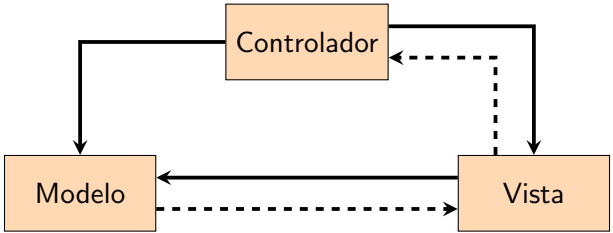
Modelo-Vista-Controlador



Vista:

- La vista es la frontera de comunicación con el mundo exterior.
- La vista está expuesta al usuario.
  - El usuario interactúa con la vista para lograr sus metas.
  - La vista notifica al controlador.
  - El controlador toma las acciones que corresponda.
  - El modelo, de cambiar, notifica a la vista de manera de que actualice su estado.

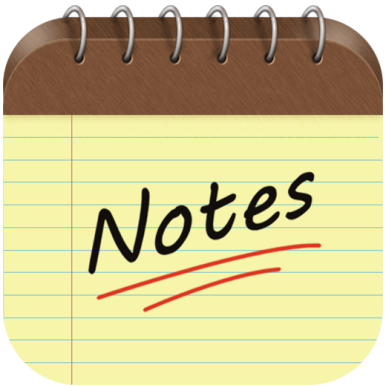
Modelo-Vista-Controlador



Ventajas:

- Es un buen punto de partida para organizar nuestro código.
- Es fácil cambiar la vista sin modificar el controlador o el modelo.
- Se pueden tener varias vistas (e.j., vista mobile o web).
- El mismo modelo puede ser usado por varias aplicaciones.

Ejemplo: Notas



Discutamos un ejemplo

Ejemplo: Notas

```
¡Bienvenido!

¿Qué quieres hacer?
0- Mostrar notas.
1- Agregar una nueva nota.
2- Guardar notas.
3- Salir.
0
-------------- Estas son mis notas actuales

--------------

¿Qué quieres hacer?
0- Mostrar notas.
1- Agregar una nueva nota.
2- Guardar notas.
3- Salir.
```

Ejemplo: Notas

```
1
Ingresa tu nota:
Recuerda resolver la E2.

¿Qué quieres hacer?
0- Mostrar notas.
1- Agregar una nueva nota.
2- Guardar notas.
3- Salir.
1
Ingresa tu nota:
Prepara los documentos del comité de ética.

¿Qué quieres hacer?
0- Mostrar notas.
1- Agregar una nueva nota.
2- Guardar notas.
```

Ejemplo: Notas

```
3- Salir.
2
Tus notas fueron guardadas exitosamente!

¿Qué quieres hacer?
0- Mostrar notas.
1- Agregar una nueva nota.
2- Guardar notas.
3- Salir.
3

¡Hasta pronto!
```

Ejemplo: Notas

```
3- Salir.
2
Tus notas fueron guardadas exitosamente!

¿Qué quieres hacer?
0- Mostrar notas.
1- Agregar una nueva nota.
2- Guardar notas.
3- Salir.
3

¡Hasta pronto!
```

Ahora podemos volver a abrir la aplicación y ver que nuestras notas fueron guardadas.

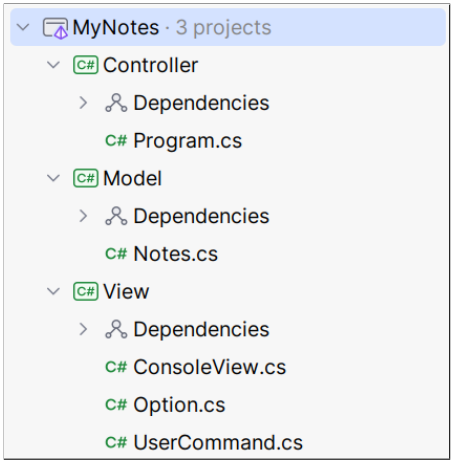
Ejemplo: Notas

```
¡Bienvenido!

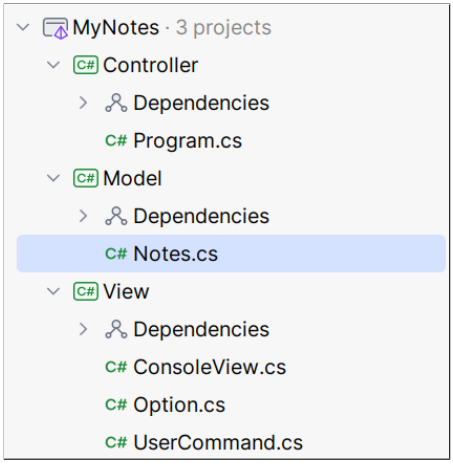
¿Qué quieres hacer?
0- Mostrar notas.
1- Agregar una nueva nota.
2- Guardar notas.
3- Salir.
0
-------------- Estas son mis notas actuales
Recuerda resolver la E2.
Prepara los documentos del comité de ética.
--------------

¿Qué quieres hacer?
0- Mostrar notas.
1- Agregar una nueva nota.
2- Guardar notas.
```

Ejemplo: Notas



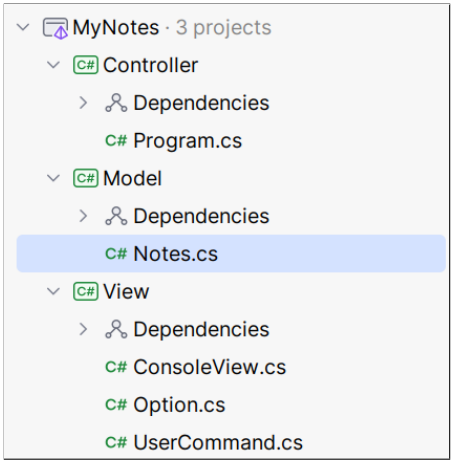
Ejemplo: Notas



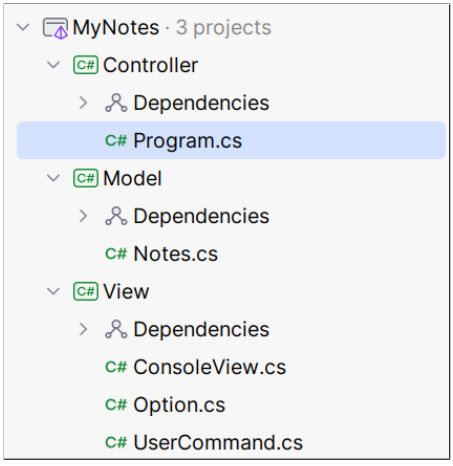
Ejemplo: Notas

```
1  public class Notes{
2    private const string NotesFile = "my_notes.txt";
3    private List<string> _myNotes = new ();
4
5    public Notes() {
6      if(File.Exists(NotesFile))
7        _myNotes = File.ReadAllLines(NotesFile).ToList();
8    }
9
10   public void AddNote(string newNote)
11     => _myNotes.Add(newNote);
12
13   public void SaveNotes()
14     => File.WriteAllLines(NotesFile, _myNotes);
15
16   public override string ToString()
17     => string.Join("\n", _myNotes);
18 }
```

Ejemplo: Notas



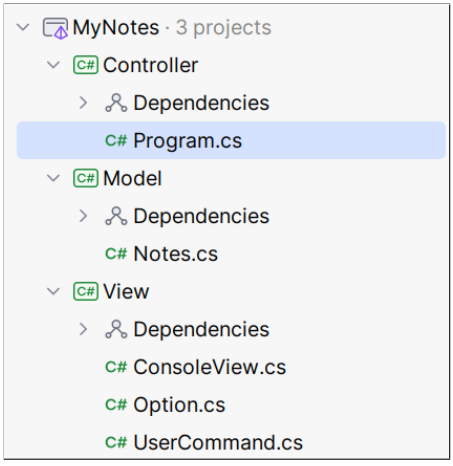
Ejemplo: Notas



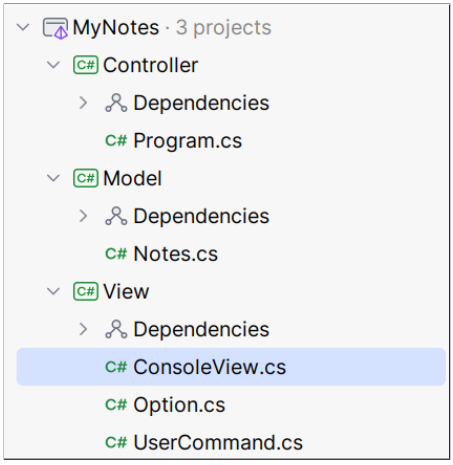
Ejemplo: Notas

```
4  Notes notes = new Notes();
5  ConsoleView view = new ConsoleView(notes);
6
7  view.ShowWelcomeMessage();
8  UserCommand command;
9  do {
10   command = view.AskUserWhatToDoNext();
11   if (command == UserCommand.ShowNotes)
12     view.ShowNotes();
13   if (command == UserCommand.Save) {
14     notes.SaveNotes();
15     view.AnnounceNotesWereSaved();
16   }
17   if (command == UserCommand.AddNote) {
18     string newNote = view.AskUserToInputANote();
19     notes.AddNote(newNote);
20   }
21 } while (command != UserCommand.Exit);
22 view.SayGoodbye();
```

Ejemplo: Notas



Ejemplo: Notas



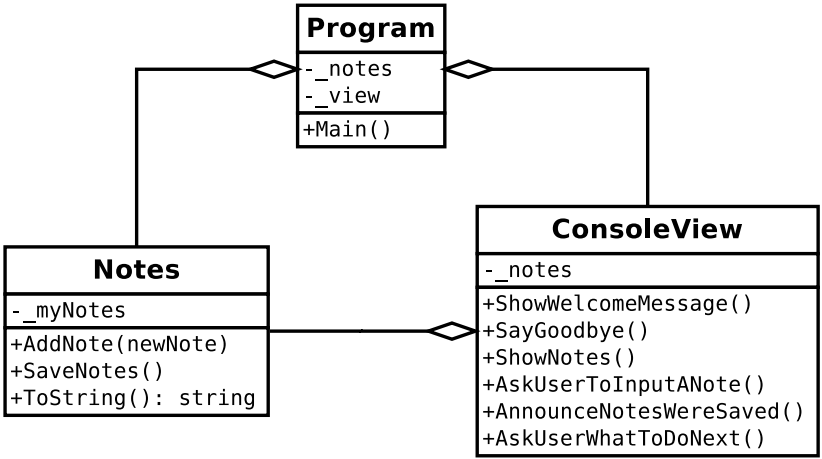
Ejemplo: Notas

```
1  public class ConsoleView {
2    private Notes _notes;
3    private List<Option> _options;
4
5    public ConsoleView(Notes notes) {
6      _notes = notes;
7      LoadOptions();
8    }
9
10   private void LoadOptions() { /* ... */ }
11
12   public void ShowWelcomeMessage()
13     => Console.WriteLine("\n¡Bienvenido!");
14
15   public void SayGoodbye()
16     => Console.WriteLine("\n¡Hasta pronto!");
17
18   public void ShowNotes() {
19     Console.WriteLine("-------------- Estas son mis notas actuales");
20     Console.WriteLine(_notes.ToString());
21     Console.WriteLine("--------------");
22   }
```

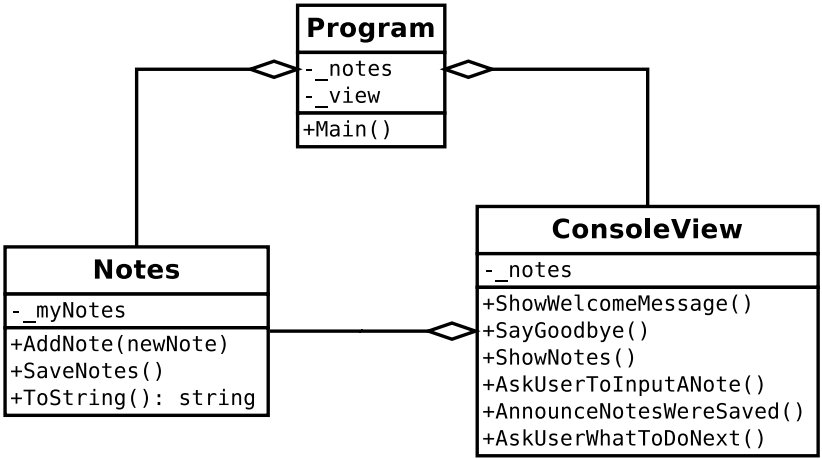
Ejemplo: Notas

```
24    public string AskUserToInputANote() {
25      Console.WriteLine("Ingresa tu nota: ");
26      return Console.ReadLine();
27    }
28
29    public void AnnounceNotesWereSaved()
30      => Console.WriteLine("Tus notas fueron guardadas exitosamente!");
31
32    public UserCommand AskUserWhatToDoNext() {
33      Console.WriteLine("\n¿Qué quieres hacer?");
34      ShowOptions();
35      int optionId = SelectOptionNumber(_options.Count - 1);
36      return _options[optionId].Command;
37    }
38
39    private void ShowOptions() { /* ... */ }
40    private int SelectOptionNumber(int max) { /* ... */ }
41  }
```

Ejemplo: Notas



Ejemplo: Notas



... a simple vista, esta es la idea más tonta que he visto.

Ejemplo: Notas

¿Alguien me puede explicar como es que esto

```
Notes notes = new Notes();
ConsoleView view = new ConsoleView(notes);
view.ShowWelcomeMessage();
// ... etc
```

```
public class ConsoleView {
  /* ... */
  public void ShowWelcomeMessage()
    => Console.WriteLine("\n¡Bienvenido!");
}
```

Ejemplo: Notas

¿Alguien me puede explicar como es que esto

```
Notes notes = new Notes();
ConsoleView view = new ConsoleView(notes);
view.ShowWelcomeMessage();
// ... etc
```

```
public class ConsoleView {
  /* ... */
  public void ShowWelcomeMessage()
    => Console.WriteLine("\n¡Bienvenido!");
}
```

... es mejor/más claro que simplemente hacer esto??

```
Notes notes = new Notes();
Console.WriteLine("\n¡Bienvenido!");
// ... etc
```

Ejemplo: Notas

Importante: MVC se justifica en proyectos donde la vista cambia mucho o en los que queremos ser capaces de cambiar entre varias vistas.

Ejemplo: Notas

Importante: MVC se justifica en proyectos donde la vista cambia mucho o en los que queremos ser capaces de cambiar entre varias vistas.

Escenarios posibles:

1. Queremos cambiar la vista a inglés.

Ejemplo: Notas

Importante: MVC se justifica en proyectos donde la vista cambia mucho o en los que queremos ser capaces de cambiar entre varias vistas.

Escenarios posibles:

1. Queremos cambiar la vista a inglés.
2. Queremos ser capaces de cambiar entre vista inglés y vista español.

Ejemplo: Notas

Importante: MVC se justifica en proyectos donde la vista cambia mucho o en los que queremos ser capaces de cambiar entre varias vistas.

Escenarios posibles:

1. Queremos cambiar la vista a inglés.
2. Queremos ser capaces de cambiar entre vista inglés y vista español.

¿Cómo lo hacemos?

Ejemplo: Notas

Importante: MVC se justifica en proyectos donde la vista cambia mucho o en los que queremos ser capaces de cambiar entre varias vistas.

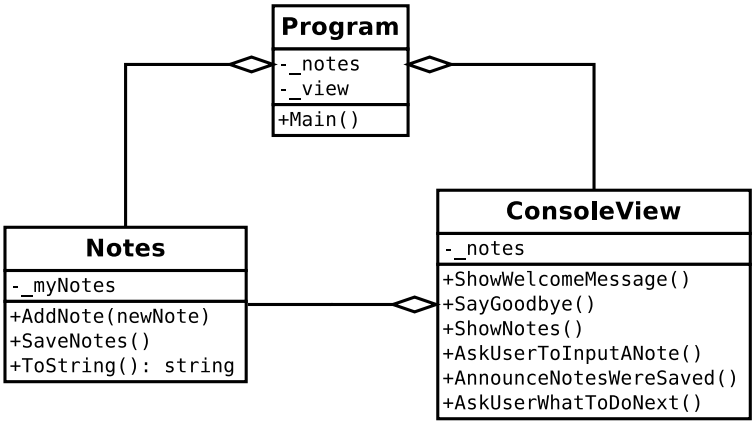
Escenarios posibles:

1. Queremos cambiar la vista a inglés.
2. Queremos ser capaces de cambiar entre vista inglés y vista español.

¿Cómo lo hacemos?

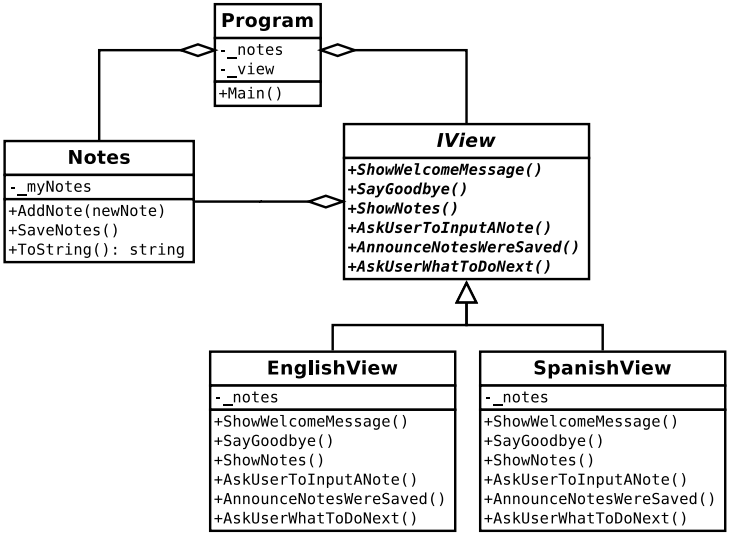
Gracias a que el proyecto está en MVC, no tenemos que cambiar ni el modelo ni el controlador. Basta con agregara "*una nueva vista*."

Ejemplo: Notas



Noten que el controlador funcionará con cualquier objeto que implemente los mismos métodos públicos que tiene `ConsoleView`.

Ejemplo: Notas



Ejemplo: Notas

```
1 public interface IView {
2   public void ShowWelcomeMessage();
3   public void SayGoodbye();
4   public void ShowNotes();
5   public string AskUserToInputANote();
6   public void AnnounceNotesWereSaved();
7   public UserCommand AskUserWhatToDoNext();
8 }
```

Ejemplo: Notas

```
1 public interface IView {
2   public void ShowWelcomeMessage();
3   public void SayGoodbye();
4   public void ShowNotes();
5   public string AskUserToInputANote();
6   public void AnnounceNotesWereSaved();
7   public UserCommand AskUserWhatToDoNext();
8 }
```

```
1 public class SpanishView : IView {
2   /* ... */
3   public void ShowWelcomeMessage()
4     => Console.WriteLine("\n¡Bienvenido!");
5   public void SayGoodbye()
6     => Console.WriteLine("\n¡Hasta pronto!");
7   /* ... */
8 }
```

Ejemplo: Notas

```
1 public interface IView {
2   public void ShowWelcomeMessage();
3   public void SayGoodbye();
4   public void ShowNotes();
5   public string AskUserToInputANote();
6   public void AnnounceNotesWereSaved();
7   public UserCommand AskUserWhatToDoNext();
8 }
```

```
1 public class EnglishView : IView {
2   /* ... */
3   public void ShowWelcomeMessage()
4     => Console.WriteLine("\nWelcome!");
5   public void SayGoodbye()
6     => Console.WriteLine("\nBye bye!");
7   /* ... */
8 }
```

Ejemplo: Notas

```
Notes notes = new Notes();

IView view;
if(Configuration.Language == "English")
  view = new EnglishView(notes);
else if (Configuration.Language == "Spanish")
  view = new SpanishView(notes);
else
  throw new ApplicationException("Unsupported language exception");

view.ShowWelcomeMessage();
UserCommand command;
do
{
  /* ... etc ... */
} while (command != UserCommand.Exit);

view.SayGoodbye();
```

Ejemplo: Domino



Ahora discutamos un ejemplo un poco más complejo.

Ejemplo: Domino

Este proyecto permite jugar dominó en consola:

Ejemplo: Domino

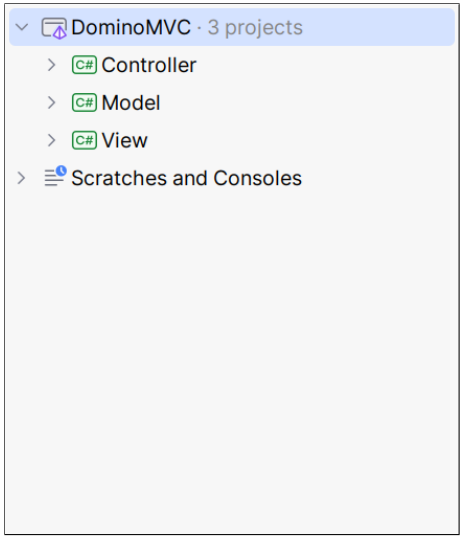
Este proyecto permite jugar dominó en consola:

```
Mano jugador 0: [2,4][5,5][1,1][1,3][1,4][3,5][2,5]
Jugador 1 tiene 7 fichas en la mano.
Jugador 2 tiene 7 fichas en la mano.
Jugador 3 tiene 7 fichas en la mano.
Jugador 3 tiene 7 fichas en la mano.
MESA:
Jugador 3 tiene 6 fichas en la mano.
Jugador juega [6,6] al inicio de la línea.

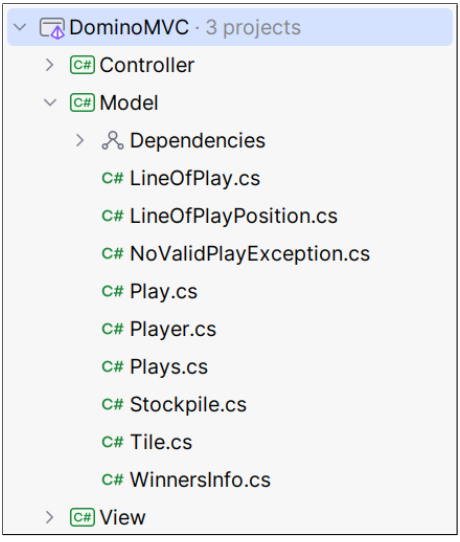
Mano jugador 0: [2,4][5,5][1,1][1,3][1,4][3,5][2,5]
MESA: [6,6]
No hay fichas que puedas jugar! :(

Jugador 1 tiene 7 fichas en la mano.
MESA: [6,6]
Jugador 1 tiene 6 fichas en la mano.
Jugador juega [3,6] al inicio de la línea.
```

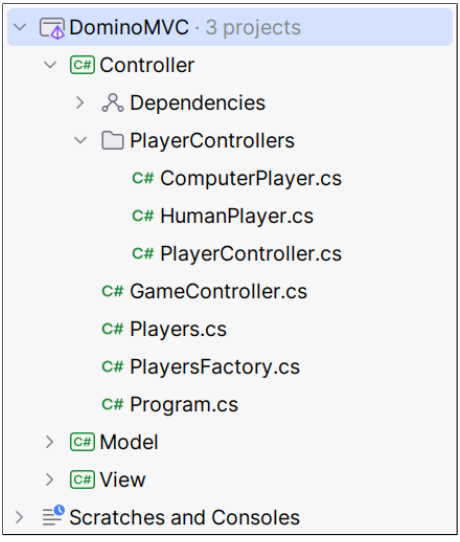
Ejemplo: Domino



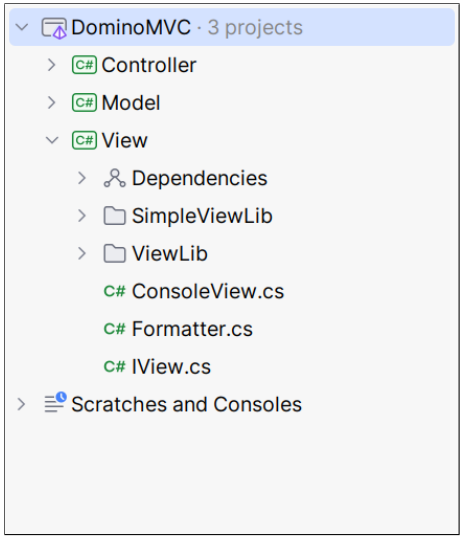
Ejemplo: Domino



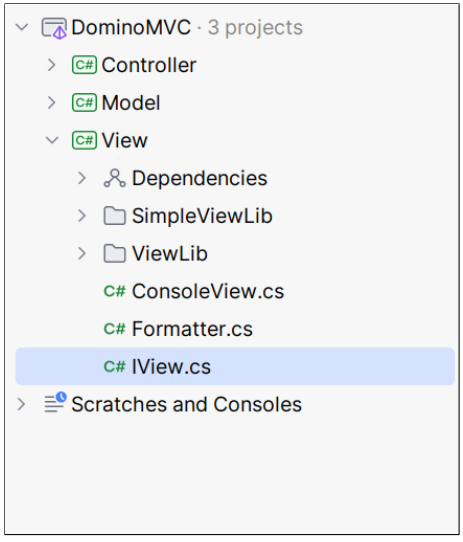
Ejemplo: Domino



Ejemplo: Domino



Ejemplo: Domino



Ejemplo: Domino

Gracias a MVC, cualquier vista que implemente esta interfaz debería funcionar:

```
public interface IView
{
  void Start(Action startProgram);
  void Show(LineOfPlay lineOfPlay);
  void ShowHand(Player player);
  void ShowHandSize(Player player);
  Play AskUserToSelectPlay(Plays validPlays);
  void CongratulateWinner(WinnersInfo winners);
  void AnnouncePlayWasSuccessful(Play play);
  void AnnouncePlayerMustPass();
  void AnnouncePlayerPassed();
}
```

Ejemplo: Domino

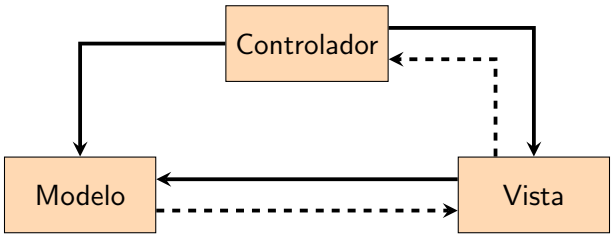
Gracias a MVC, cualquier vista que implemente esta interfaz debería funcionar:

```
public interface IView
{
  void Start(Action startProgram);
  void Show(LineOfPlay lineOfPlay);
  void ShowHand(Player player);
  void ShowHandSize(Player player);
  Play AskUserToSelectPlay(Plays validPlays);
  void CongratulateWinner(WinnersInfo winners);
  void AnnouncePlayWasSuccessful(Play play);
  void AnnouncePlayerMustPass();
  void AnnouncePlayerPassed();
}
```

Obs: Gracias a que la vista conoce al modelo podemos pasar clases completas, por ejemplo `Tile`, desde el controlador a la vista :)

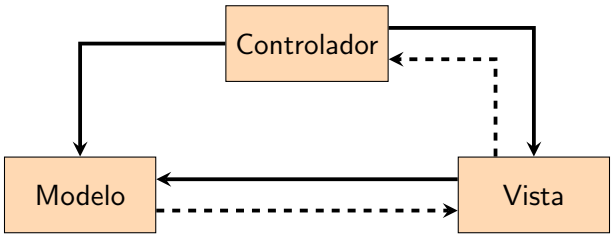
Ejemplo: Domino

Esto me recuerda, ¿cómo decido qué va en cada parte en mi E2?



Ejemplo: Domino

Esto me recuerda, ¿cómo decido qué va en cada parte en mi E2?



Reglas al ojo:

- Toda interacción con el usuario va en la vista (i.e., input/outputs y su formateo).
- Si requiere a la vista para funcionar, es un controlador.
- Si no requiere a la vista, es parte del modelo.

Ejemplo: Domino

Volvamos a esto de la interfaz:

```
1  public interface IView
2  {
3    void Start(Action startProgram);
4    void Show(LineOfPlay lineOfPlay);
5    void ShowHand(Player player);
6    void ShowHandSize(Player player);
7    Play AskUserToSelectPlay(Plays validPlays);
8    void CongratulateWinner(WinnersInfo winners);
9    void AnnouncePlayWasSuccessful(Play play);
10   void AnnouncePlayerMustPass();
11   void AnnouncePlayerPassed();
12 }
```

Ejemplo: Domino

Volvamos a esto de la interfaz:

```
1  public interface IView
2  {
3    void Start(Action startProgram);
4    void Show(LineOfPlay lineOfPlay);
5    void ShowHand(Player player);
6    void ShowHandSize(Player player);
7    Play AskUserToSelectPlay(Plays validPlays);
8    void CongratulateWinner(WinnersInfo winners);
9    void AnnouncePlayWasSuccessful(Play play);
10   void AnnouncePlayerMustPass();
11   void AnnouncePlayerPassed();
12 }
```

Obs: Para que el domino funcione con una interfaz gráfica debería basta con implementar `IView` – sin requerir hacer ningún cambio ni al controlador ni al modelo.

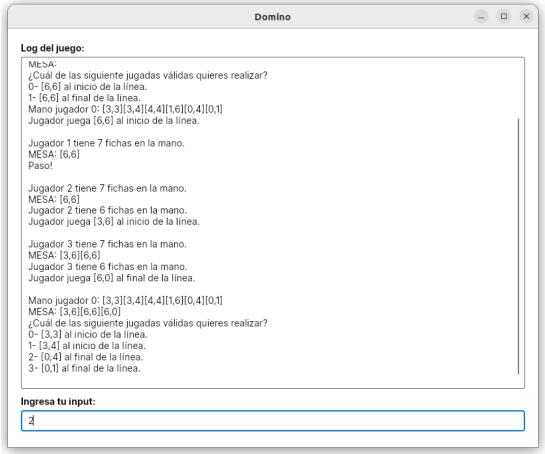
Actividad

Actividad

Implementé dos interfaces gráficas para que conecten con el proyecto DominoMVC.

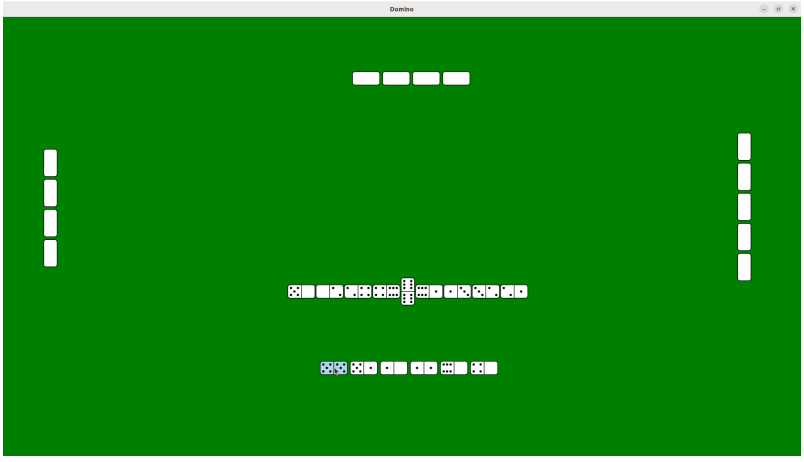
Actividad

Implementé dos interfaces gráficas para que conecten con el proyecto DominoMVC.



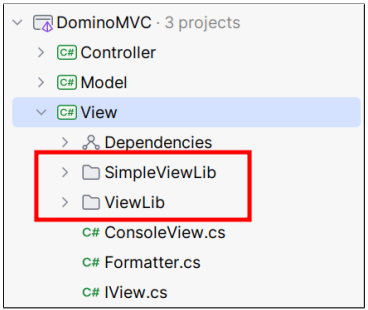
Actividad

Implementé dos interfaces gráficas para que conecten con el proyecto DominoMVC.



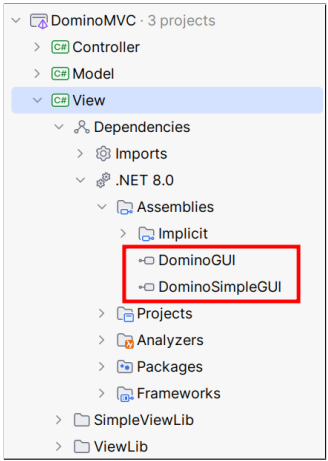
Actividad

Ambas interfaces ya están incluidas en el proyecto DominoMVC:



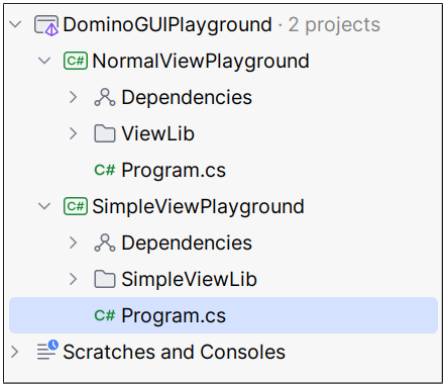
Actividad

Ambas interfaces ya están incluidas en el proyecto DominoMVC:



Actividad

El proyecto `DominoGUIPlayground` tiene ejemplos sobre cómo usar las interfaces:



Actividad

Por ejemplo, así se puede usar la primera vista:

```
using DominoSimpleGUI;

DominoWindow dominoWindow = new DominoWindow();
dominoWindow.Start(Run);

void Run() {
  dominoWindow.Write("¡Hola!");
  dominoWindow.WriteLine(" Este es un ejemplo.");
  dominoWindow.WriteLine("De lo que se puede lograr con esta vista.");
  dominoWindow.WriteLine("\nAhora ingresa un input :)");
  string input = dominoWindow.ReadLine();
  dominoWindow.WriteLine($"\nTu input fue: \"{input}\"");
  dominoWindow.WriteLine("\n¡Adiós!");
}
```

Actividad

Por ejemplo, así se puede usar la primera vista:

```
using DominoSimpleGUI;

DominoWindow dominoWindow = new DominoWindow();
dominoWindow.Start(Run);

void Run() {
  dominoWindow.Write("¡Hola!");
  dominoWindow.WriteLine(" Este es un ejemplo.");
  dominoWindow.WriteLine("De lo que se puede lograr con esta vista.");
  dominoWindow.WriteLine("\nAhora ingresa un input :)");
  string input = dominoWindow.ReadLine();
  dominoWindow.WriteLine($"\nTu input fue: \"{input}\"");
  dominoWindow.WriteLine("\n¡Adiós!");
}
```

Notar que se requiere darle el método `Run` a `Start(...)`.

Actividad

Por ejemplo, así se puede usar la primera vista:

```
using DominoSimpleGUI;

DominoWindow dominoWindow = new DominoWindow();
dominoWindow.Start(Run);

void Run() {
  dominoWindow.Write("¡Hola!");
  dominoWindow.WriteLine(" Este es un ejemplo.");
  dominoWindow.WriteLine("De lo que se puede lograr con esta vista.");
  dominoWindow.WriteLine("\nAhora ingresa un input :)");
  string input = dominoWindow.ReadLine();
  dominoWindow.WriteLine($"\nTu input fue: \"{input}\"");
  dominoWindow.WriteLine("\n¡Adiós!");
}
```

Notar que se requiere darle el método `Run` a `Start(...)`. Esto es culpa de MacOS.

Actividad

... y nos obliga a hacer eso incluso para la vista en consola (`Program.cs`):

```
using Controller;
using View;

IView view = new ConsoleView();
view.Start(Run);
void Run() {
  GameController gameController = new GameController(view);
  gameController.Play();
}
```

Actividad

... y nos obliga a hacer eso incluso para la vista en consola (`Program.cs`):

```
using Controller;
using View;

IView view = new ConsoleView();
view.Start(Run);
void Run() {
  GameController gameController = new GameController(view);
  gameController.Play();
}
```

El código de la vista en consola llama directo al método dado:

```
public class ConsoleView : IView {
  public void Start(Action startProgram)
    => startProgram();
  /* ... */
}
```

Actividad

Actividades:

- **09/09**: Agregar la vista *simple* al `DominoMVC`.
- **11/09**: Agregar vista *compleja* al `DominoMVC`

Actividad

Actividades:

- **09/09**: Agregar la vista *simple* al `DominoMVC`.
- **11/09**: Agregar vista *compleja* al `DominoMVC`

Ambas actividades se encuentran disponibles en canvas :)

Actividad

Actividades:

- **09/09**: Agregar la vista *simple* al `DominoMVC`.
- **11/09**: Agregar vista *compleja* al `DominoMVC`

Ambas actividades se encuentran disponibles en canvas :)

HINT: Para la primera actividad, basta con crea una nueva vista que implemente `IView`. Luego hay que hacer lo mismo que con la vista consola pero llamando a los métodos del write y read del `DominoWindow` en vez de a la consola.

¿Cuáles son los descuentos típicos en el proyecto debido al MVC?

Modelo-Vista-Controlador

Descuentos más comunes:

- No se distinguen que clases pertenecen a cada componente.
- El modelo es dependiente de la vista
- El modelo es dependiente del controlador
- El proyecto vista depende del controlador (genera dependencias circulares)
- Se modifican los datos del modelo directamente en la vista
- El controlador depende mucho de que la vista sea por consola (se utiliza `view.WriteLine()` o `view.ReadLine()` directamente en el controlador)