Lea atentamente

El siguiente procedimiento, describe el paso a paso que debe seguir algún alumno designado de cada grupo y contiene la información suficiente para tener un orden básico de ejecución y administración del proyecto.

El contenido de este documento es genérico y para todos los grupos y cursos igual. Cada grupo debe ir trabajando sobre cada uno de ellos, generando nuevos issue y asociándolos a su respectivo proyecto.

Todas las siguientes tareas deben ser ejecutadas desde Github, desde el repositorio del proyecto que les fue designado.

**Issues**

Al cargar cada issues debe asignarlo al proyecto (Backlog) y el mileston del lado derecho, según corresponda.

A screenshot of a computer

Description automatically generated

**Importante**: Crear todos los issues en el orden especificado para que los ids. de elloss se correlacionen con el Title y puedan ser accedidos directamente por

URL **(No generar issues de prueba, antes de cargar los 21 indicados en este documento.)**

* **Desde Github > Issues > New issue**
* Completar todos los datos indicados en la tabla (una fila por issue) > Submit new issue

|  |  |  |
| --- | --- | --- |
| **Title** | **Comment** | **Milestone** |
| 01.Planificación | - [ ] Definir que integrante es  1️⃣ Eugenio Sainte Marie  Modelos  - [ ] Definición de clases a crear  - [ ] Repartirse las clases por integrante  Propiedades  - [ ] Identificar relación y cardinalidad entre clases (idea inicial)  - [ ] Definir dependencias (Entidades fuertes y débiles) | EP1-Modelos |
| 02.Creación de proyecto | Integrante 1️⃣ con participación/presencia del equipo:  - [ ] Crear branch develop basada en main  - [ ] Checkout en develop  - [ ] Generar proyecto en carpeta clonada del proyecto  - [ ] Commit con mensaje "Creación de proyecto base"  - [ ] Push al repositorio en branch develop  Integrantes 2️⃣3️⃣ y 4️⃣  - [ ] Fetch  - [ ] Checkout develop | EP1-Modelos |
| 03.Generación de Modelos | Integrante 2️⃣ con participación/presencia del equipo:  - [ ] Crear branch ClasesBase basada en develop  - [ ] Checkout en ClasesBase  - [ ] Crear Clases base (sin propiedades) en carpeta Models  - [ ] Commit con mensaje "Generación de clases base"  - [ ] Push al repositorio en branch ClasesBase  - [ ] Checkout en develop  - [ ] Pull (para acostumbrarnos)  - [ ] Seleccionar branch ClasesBase y presionar Merge into Current Branch (que debería ser develp)  - [ ] Push  Integrantes 1️⃣ 3️⃣ y 4️⃣  - [ ] Pull desde develop | EP1-Modelos |

|  |  |  |
| --- | --- | --- |
| 04.Propiedades | Cada integrante debe trabajar sobre sus propias clases:  - Declarar propiedad Id  - Declarar propiedades adicionales  - Declarar tipos de dato para cada propiedad  - Commit con Mensaje "Propiedades para clases xxx,yyy,zzz"  \_ Nota: Se recomienda que, en esta instancia, aún no trabajen más de un integrante sobre una misma clase, para no tener problemas de conflictos al momento de hacer un push.  - [ ] Crear Branch <NombreAlumno> basada en develop  - [ ] Checkout <NombreAlumno>  - [ ] Crear propiedades autoimplementadas para las clases asignadas para cada uno  - [ ] Commit "Propiedades para clases xxx,yyy,zzz"  - [ ] Push  - [ ] Checkout develop  - [ ] Pull  - [ ] Merge <NombreAlumno> into Current Branch (develop)  - [ ] Push  \*\*Todos los integrantes:\*\*  - [ ] Una vez que terminaron todos de hacer merge, hacen Pull y deberían todos tener el mismo código.  Integrante 3️⃣ con participación/presencia del equipo:  - [ ] Checkout main  - [ ] Pull  - [ ] Merge develop into main  - [ ] Push  \*\*Todos los integrantes:\*\*  - [ ] Checkout main  - [ ] Pull  https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/auto-implemented-properties | EP1-Modelos |

|  |  |  |
| --- | --- | --- |
| 05.Planificación | Todos los integrantes deben aplicar sobre sus clases:  - [ ] Revisión, Distribuir tareas y remediar según Feedback anterior  - [ ] Mejoras en las relaciones entre clases  - [ ] Propiedades relacionales  - [ ] Clases intermedias (si fuesen necesarias)    Definen en conjunto cuestiones generales de Restricciones (ValidationAttributes)  - [ ] Requeridos  - [ ] Valores Largos, Rangos, Patrones, etc.  - [ ] Mensajes de error (cada validación debe tener su mensaje de error en español) | EP2-Relaciones y Restricciones |
| 06.Tareas comunes | Integrante 4️⃣ con participación/presencia del equipo:  - [ ] Checkout develop y pull  - [ ] Genera clases adicionales (Detectadas o Solicitadas)  - [ ] Clases de soporte (Helpers,Configs, Messages,Alias, etc.)  - [ ] Configuraciones comunes (Nullable, ImplicitUsings)  - [ ] Commit "Configuraciones base del EP2"  - [ ] Push  Todos los integrantes:  - [ ] Merge develop in <NombreAlumno> branch | EP2-Relaciones y Restricciones |
| 07.Ejecución de Mejoras en Relaciones y Restricciones | Todos los integrantes aplican mejoras definidas previamente, cada uno en las clases bajo su responsabilidad para este entregable.  Aplican Restricciones (ValidationAttributes)  - [ ] Requeridos  - [ ] Valores Largos, Rangos, Patrones, etc.  - [ ] Mensajes de error (cada validación debe tener su mensaje de error en español) | EP2-Relaciones y Restricciones |

|  |  |  |
| --- | --- | --- |
| 08.Planificación | \*\*Todos los integrantes:\*\*  - [ ] Revisión, Distribuir tareas  - [ ] Remediar según Feedback anterior  - [ ] Imaginar/Charlar interacción del usuario con el sistema - modalidad de uso, asistente (Wizard, paso a paso), AIO (All-In-One). Básicamente, si se quiere que el usuario cargue todo en un único formulario de una vez cuando sea posible, o el sistema lo va guiando por pantallas, carga datos, siguiente, así sucesivamente. No obstante, considerar las funcionalidades que se piden en el enunciado. Si indica modo wizard, debe respetarse.  - [ ] Definición inicial de interfaz, componentes y texto participante  - [ ] Repartirse tareas | EP3-ABMs |
| 09.Resolución de issues | Todos los integrantes, ejecutarán las tareas de remediación planificadas en base al feedback. | EP3-ABMs |
| 10.Tareas comunes | Integrante 3️⃣ con participación/presencia del equipo en develop:  - [ ] Dependencias (Cargar las dependencia de paquetes solicitadas)  - [ ] EntityFrameworkCore  - [ ] EntityFrameworkCore.Relational  - [ ] EntityFrameworkCore.InMemory  - [ ] EntityFrameworkCore.SqlServer  - [ ] EntityFrameworkCore.Tools  - [ ] AspNetCore.Identity.EntityFrameworkCore  - [ ] Web.CodeGeneration.Design  - [ ] Tareas y configuraciones comunes Carpeta Data / EntityFramework / Servicios  - [ ] Commit "Configuraciones base EP3”  - [ ] Push | EP3-ABMs |
| 11.Creación de ABMs | Todos los integrantes generan las mejoras definidas y creación de los ABMs de las clases bajo su responsabilidad en su rama.  - [ ] Merge a develop 1️⃣  - [ ] Merge a develop 2️⃣  - [ ] Merge a develop 3️⃣  - [ ] Merge a develop 4️⃣ | EP3-ABMs |

|  |  |  |
| --- | --- | --- |
| 12.Planificación | Todos los integrantes:  - [ ] Revisión, Distribuir tareas y remediar según Feedback anterior  - [ ] Revisión y definición de prioridades en las funcionalidades  - [ ] Repartirse tareas | EP4-Funcionalidades básicas |
| 13.Resolucion de issues | Todos los integrantes, ejecutarán las tareas de remediación planificadas en base al feedback.  - [ ] Integrante 1️⃣  - [ ] Integrante 2️⃣  - [ ] Integrante 3️⃣  - [ ] Integrante 4️⃣ | EP4-Funcionalidades básicas |
| 14.Tareas comunes | Integrante 2️⃣ con participación/presencia del equipo:  - [ ] Tareas y configuraciones comunes - Identity  - [ ] Herencia de Persona es un IdentityUser  - [ ] Creación de Rol, si fuese deseado  - [ ] Configuración de servicio Identity  - [ ] Commit "Configuraciones base EP4"  - [ ] Push | EP4-Funcionalidades básicas |
| 15.Identity management | Agregar las siguientes funcionalidades:  Usuarios:  - [ ] Registración (auto-registración para entidad que corresponda, si es que corresponde)  - [ ] Login  - [ ] Logout  En ABMs ya existentes relacionados con IdentityUser:  - [ ] Adecuar Create  - [ ] Adecuar Edit  Roles - Modificaciones e implicancias en:  - [ ] En registración (asignación de roles automático, dependiendo la necesidad)  - [ ] En Creación (asignación de roles automático, dependiendo la necesidad)  - [ ] En Acceso (Restringir acceso basado en roles)  - [ ] En Layout (Adecuar el layout, para mostrar diferentes funcionalidades de acceso, basado en rol) | EP4-Funcionalidades básicas |
| 16.Funcionalidades básicas | Completar funcionalidades básicas definidas en la planificación, según orden de prioridad.  Recordar que falta de validaciones o errores en consistencias de datos, pueden ser permitidos en esta instancia. Ej. Poder alquilar una cochera, cuando todas las cocheras ya están tomadas. | EP4-Funcionalidades básicas |

|  |  |  |
| --- | --- | --- |
| 17.Planificación | Todos los integrantes:  - [ ] Revisión, Distribuir tareas y remediar según Feedback anterior  - [ ] Revisión y definición de prioridades en las funcionalidades  - [ ] Repartirse tareas  - [ ] Planificación inicial de Presentación (si fuese requerido) | TP Finalizado |
| 18.Resolución de issues | Todos los integrantes, ejecutarán las tareas de remediación planificadas en base al feedback.  - [ ] Integrante 1️⃣  - [ ] Integrante 2️⃣  - [ ] Integrante 3️⃣  - [ ] Integrante 4️⃣ | TP Finalizado |
| 19.Ajustes de Funcionalidades básicas | Realizar ajustes a las funcionalidades básicas definidas en la planificación según orden de prioridad.  Recordar que para aprobar el proyecto durante la cursada se requiere el ~80% de las funcionalidades completadas. | TP Finalizado |
| 20.Funcionalidades completas | Completar la totalidad de funciones solicitadas. | TP Finalizado |
| 21.Checklist de evaluación | La descripción de este issue, será editado por el docente según cada proyecto y dispondrán del checklist correspondiente. | TP Finalizado |

**Al finalizar la tarea deberían quedar:**

|  |  |
| --- | --- |
| Milestones |  |
| Issues |  |
| Project |  |