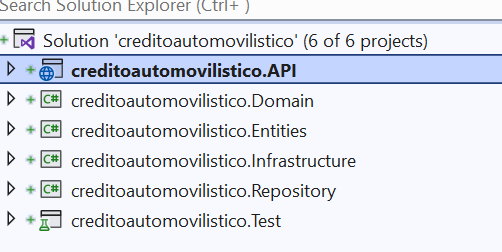
**Onboarding Practico**

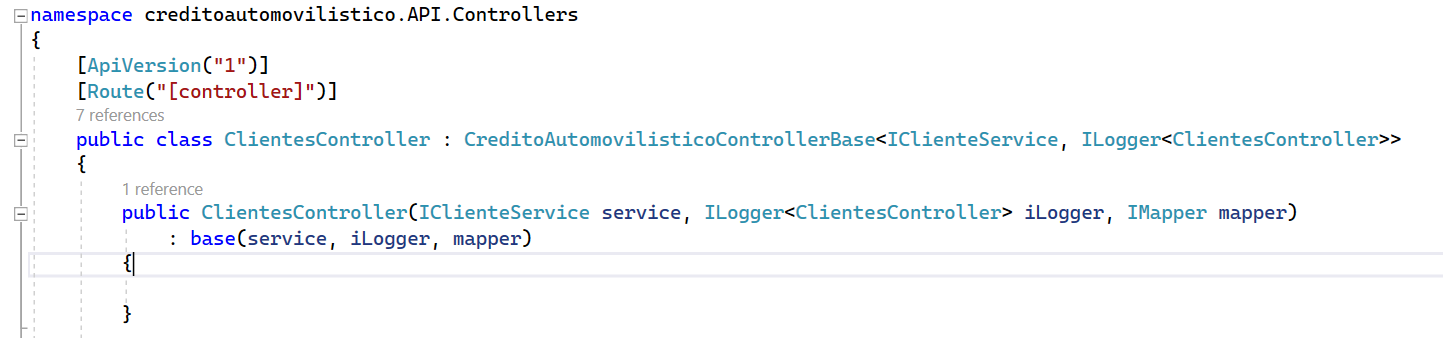
|  |  |
| --- | --- |
| **Información del Recurso** | |
| **Perfil Back o Front** | Back |
| **Nombre:** | Jorge Enrique Ulloa Guardarramos |
| **Email personal:** | Jorge.ulloa@globant.com |
| **Email banco:** | jeulloa@pichincha.com |
| **Computador (código)** |  |
| **Tribu o Célula** | TEAM HABILITADOR DE NEGOCIO -CÉLULA: CELULA BUSINESS CAPABILITY CANALES Y PASIVOS |
| **Información del Documento** | |
| **Tutor:** | Cristian Yepez |
| **Fecha Inicio OnBoarding** | 15/08/2022 |
| **Fecha Fin Onboarding** | 22/08/2022 |

# Informe de parte de Colaborador

* Se realizó un desarrollo basado en la arquitectura propuesta en el archivo de solución, con nombre creditoautomovilistico.



* Se creó la BDD en PostgreSQL con EF 6 Core, usando code-first migration. Se crearon los models y el databasecontext en el proyecto de Infrastucture.
* Se efectuó el seed inicial de los datos de Ejecutivos, Patios, Clientes y Marcas a partir de archivos .cvs usando CvsHelper. Esta carga se efectúa una sola vez, chequeando al abrir la aplicación si ya existen datos en la BDD.
* Se implenetaron los UnitTests para realizar el desarrollo a partir de TDD. Para los UT empleamos Nunit, se crearon clases de Mock y se usó Nsubstitute. Se crearon tres clases para UnitTests (Clientes, Patio y Vehículos).
  + Se creaton tests para cada caso de uso contra el controller, usando mocks y Nsubstitute para manejar las repuestas de los repos.
  + Una vez que la versión inicial de los tests falló, se fueron haciendo afinamientos progresivos y refactorizaciones, buscando un mayor grado de completamiento del desarrollo en cada paso.
* Se desarrollaron los endpoints requeridos para los controlladores de Clientes, Patios y Vehiculos. Las clases de controllers heredan de CreditoAutomovilisticoControllerBase y reciben por inyección de dependencias un Imapper,un Ilogger y una interfaz de tipo I<Controller>Service, definida en el proyecto de tipo Domain. En la definición de la clase CreditoAutomovilisticoControllerBase se emplea Generics para recibir el tipo de Service e Ilogger.



* Para todos los controllers se implementaron las acciones de Get, Post, Put y Delete. Adicionalmente, en el controller de Clientes se incluyó un action para generar una solicitud de crédito.
* El flujo de un request quedaría como sigue:

Controller (en layer API)

mapea los modelos del request y el respose hacia y desde los modelos de Entities

→ Iservice (en layer Domain)

→ Service (en layer Domain)

incluye las reglas de negocio

→ Irepository (en layer Domain)

→ Repository (en layer Repository)

mapea los modelos de Entities hacia y desde los modelos del context (BDD)

implementa el acceso al context (BDD)

# Feedback acerca del ejercicio

* El ejercicio se hace largo, teniendo en cuenta la gran cantidad de código que conlleva la arquitectura DDD (en mi caso fueron 86 clases). Es muy difícil implementar adecuadamente todos los casos de uso. Creemos que si el objetivo principal es valorar las habilidades en cuanto a TDD y familiarizar a las nuevas incorporaciones con la arquitectura empleada en Pichincha, existen varias opciones a partir de las cuales se puede mejorar:
  + Eliminar la necesidad de hacer una carga a partir de datos en csv. Este no es un problema trivial y tiene varias maneras de hacerlo, a cualquier desarrollador le ha de tomar varias horas la solución si no lo ha hecho antes, por tanto, si esta habilidad no es algo que se está buscando medir específicamente con el ejercicio, podría ser excluida (por demás, existe una dependencia entre Ejecutivo y Patio, por tanto, si se mantiene la necesidad de efectuar la carga inicial, debe incluirse también la tabla de Patios entre las cargadas).
  + Disminuir el scope del ejercicio (por ejemplo, especificando que no es necesario efectuar las implementaciones de acceso a datos en el repositorio).
  + Hacer un ejercicio más corto (por ejemplo, con un solo controlador y máximo dos entidades) de modo que los desarrolladores no se sientan presionados por el tiempo y puedan implementar a conciencia todas las mejores prácticas y técnicas que conocen.

# Observaciones Generales

* Se debe detallar estado actual de colaborador versus el estado inicial luego del Onboarding.
* Se debe detallar los GAPS, o los puntos de mejora.

# Compromiso

* Definir fechas para monitorear los puntos de mejora.