

Seminario de actualización en programación

Trabajo práctico N°4

**VINF010679** - Carlos David Gonzalez Nexans

**VINF09222** - Gaspar Oddovero Herrera

27 de Noviembredel 2023

## Índice

[**Índice 2**](#_1fob9te)

[**Diagrama de Entidad Relación 3**](#_2et92p0)

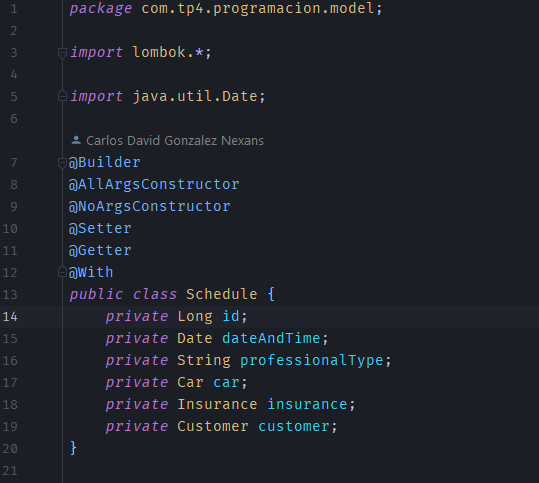
[**Patrones de diseño 3**](#_tyjcwt)

[**Estructura de proyecto entregado 4**](#_3dy6vkm)

# 

## Utilización de ORM

Se instaló y configuró el ORM Hibernate y se crearon las clases Entity que permiten abstraer los objetos de negocio de los de persistencia.

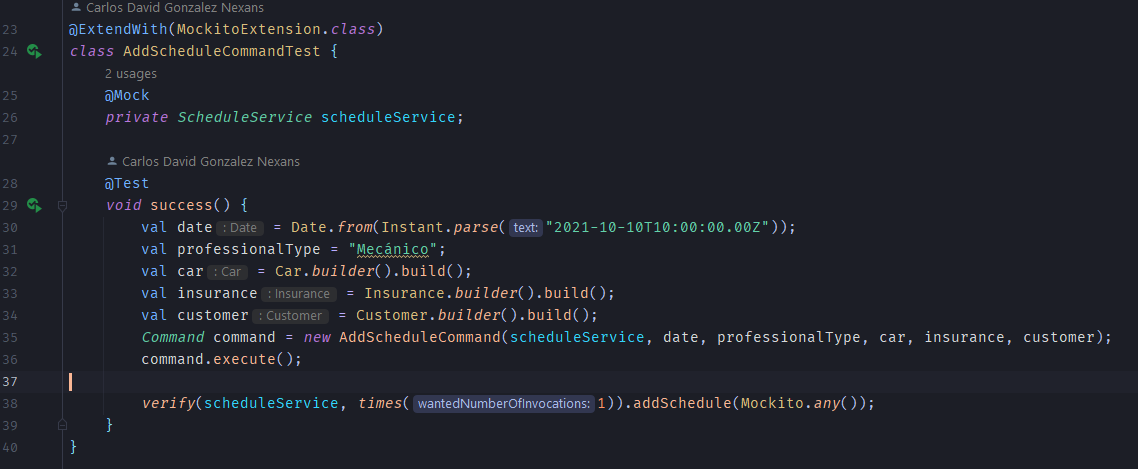


Se implementó el repositorio de datos usando el ORM, configurado para MySQL. La interfaz es genérica, por lo cual el programa funciona correctamente sustituyendo una base de datos por otra sin cambiar el código relacionado al negocio.



## Tests Unitario

Con las herramientas Junit5 y Mockito se generaron test unitarios relacionados al código de negocio. Junit permite ejecutar tests y viene con una batería de herramientas para escribirlos y Mockito permite generar clases Stub que implementen las interfaces.



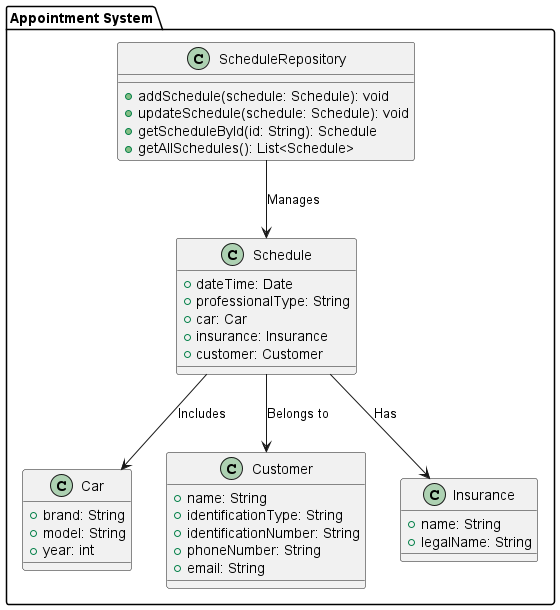
## Test de Integración

Se utilizó TestFX para levantar la aplicación y manipularla de forma automática para generar casos de prueba. Esta librería combina perfectamente con Junit.

## 

## 

## Diagrama de Entidad Relación



## Patrones de diseño

**Command** permite desacoplar la vista de la lógica de negocio, encapsulando un proceso

* AddScheduleCommand
* ChangeScheduleDateTimeCommand
* Command

**Factory** provee una interface de método que permite crear clases, permite a la vista crear los comandos

* AddScheduleCommandFactory
* ChangeScheduleDateTimeCommandFactory
* CommandFactory

**Singleton** fuerza la creación de una sóla instancia de la clase en la aplicación

* JdbcScheduleRepository
* ListScheduleRepository

**DependencyInversion** las clases que dependen de otras clases para funcionar, delegan la responsabilidad de generar estas dependencias al caller

* AddScheduleCommand
* ChangeScheduleDateTimeCommand

**Segregación de interfaces** las clases que dependen de otras clases, lo hacen desde una interfaz, permitiendo que estas clases puedan existir y evolucionar de forma independiente

* ScheduleService
* HibernateScheduleRepository
* JdbcScheduleRepository
* ListScheduleRepository

## Estructura de proyecto entregado

