

**Escuela Superior Politécnica del Litoral**

**Facultad de Electricidad y Computación**

**Diseño de Software**

**Proyecto**

*Proyecto Final*

Docente:

MSc. David Jurado Mosquera

Integrantes:

|  |  |
| --- | --- |
| NOMBRE | CORREO |
| Garcia Anchundia, Yuleixi Katherine | ykgarcia@espol.edu.ec |
| Loor Toala ,Juan Jose | @espol.edu.ec |
| Siavichay Pluas, Henry Steven | @espol.edu.ec |

2019-2S

**Tabla de contenido**

[Definición del proyecto 3](#_Toc31580573)

[Antecedentes del proyecto 3](#_Toc31580574)

[Especificaciones del proyecto 4](#_Toc31580575)

[Propuesta 7](#_Toc31580576)

[Diagrama de Casos de Uso 7](#_Toc31580577)

[Patrones de Diseño 10](#_Toc31580578)

[Builder: 10](#_Toc31580579)

[Decorator: 10](#_Toc31580580)

[Singleton: 10](#_Toc31580581)

[Diagrama de clase 11](#_Toc31580582)

[Diagramas de Secuencia 12](#_Toc31580583)

[Diagrama Lógico de la Base de Datos 18](#_Toc31580584)

[Prototipo 19](#_Toc31580585)

[Docker 19](#_Toc31580586)

[SonarQube 19](#_Toc31580587)

[Unit Test 20](#_Toc31580588)

[Referencias 21](#_Toc31580589)

**Índice de Ilustraciones**

[Ilustración 1. Diagrama de Casos de Uso 7](file:///C:\Users\Tania\Documents\yuly\diseño%20de%20software\proyecto2\informe.docx#_Toc31580559)

[Ilustración 2. Diagrama de Clases. 11](file:///C:\Users\Tania\Documents\yuly\diseño%20de%20software\proyecto2\informe.docx#_Toc31580560)

[Ilustración 3. Vendedor Genera Reporte 12](#_Toc31580561)

[Ilustración 4. Cliente envía diseño de casa 13](#_Toc31580562)

[Ilustración 5. Diagrama Lógico de la Base de Datos. 14](file:///C:\Users\Tania\Documents\yuly\diseño%20de%20software\proyecto2\informe.docx#_Toc31580563)

[Ilustración 6. Reporte SonarQube. 15](#_Toc31580564)

**Índice de Tabla**

[Tabla 1. Especificación de Caso de Uso "Diseñar Casa". 8](#_Toc31580565)

[Tabla 2.Especificaion de Caso de Uso "Realizar Diseño de Casa". 9](#_Toc31580566)

# Definición del proyecto

## Antecedentes del proyecto

La empresa MyHome S.A. es una constructora que solicita un sistema para diseñar casas para sus clientes potenciales. Este sistema debe funcionar como un quiosco que se va a ubicar dentro de las oficinas de la empresa.

Al equipo de desarrollo se le pedide que realice el diseño del sistema, incluyendo el diseño normalizado del modelo físico de la base de datos y los diagramas de clases, de casos de uso, y de secuencias utilizando la arquitectura MVC. Además, se deben aplicar patrones de diseño, presentar un prototipo funcional, usar SonarQube para analizar la calidad del código, Docker como contenedor de la base de datos y las pruebas unitarias correspondientes.

## Especificaciones del proyecto

El sistema debe tener una interfaz gráfica sencilla de utilizar en donde, de acuerdo con el rol de cada usuario se le debe mostrar la interfaz correspondiente. El sistema debe cumplir con las siguientes características:

1. Actualmente existe una sola oficina, sin embargo, a futuro se planea colocar quiscos en algunos centros comerciales para atraer a la mayor cantidad de clientes potenciales. Por lo tanto, el sistema debe conectarse a una base de datos central, donde se también se conectará otro sistema para generar reportes del uso del sistema. La base de datos debe estar almacenada en un contenedor Docker para realizar fácilmente las pruebas del sistema.
2. El sistema debe permitir el ingreso a los clientes registrados, administrador y vendedores. Todos pueden acceder con sus credenciales previamente almacenadas, pero también se les permite el acceso a los nuevos clientes a ciertas funcionalidades, siempre y cuando al final del proceso se le pida que para completar dicho proceso debe registrarse.
3. Un cliente nuevo puede realizar el diseño de una casa nueva, consultar características de esta, ver presupuesto inicial de la casa básica y registrarse. Para registrarse debe ingresar sus nombres, apellidos, cédula o pasaporte, número de celular, correo electrónico y dirección domicilio, dirección del trabajo, empresa donde trabaja, cargo, teléfono del trabajo, estado civil, número de hijos.
4. Un empleado (vendedor o administrador) solo puede ser registrado por un administrador y debe tener los siguientes datos: nombres, apellidos, cédula o pasaporte, número de celular, correo electrónico y dirección domicilio, teléfono del trabajo, estado civil, cargo (vendedor o administrador). Los administradores pueden realizar todas las operaciones CRUD sobre los datos del sistema, a excepción de la eliminación, ya que esta sólo puede ser de tipo lógica para evitar perdida de información (Manejar un campo boolean como activo o inactivo para la eliminación).
5. Un vendedor puede revisar los datos de los clientes y las casas que ha diseñado, además puede generar un reporte en PDF que el sistema enviará a su correo con los datos del cliente seleccionado. Dicho reporte debe contener los datos del cliente y los datos de las casas prediseñadas, incluyendo precios y características. También, el vendedor puede enviarle una casa prediseñada (en PDF) a un cliente específico por correo electrónico. El vendedor, también puede diseñar una casa y almacenarla para poder compartirla con sus clientes.
6. El cliente registrado puede diseñar una casa a su gusto, pero debe elegir de las casas básicas y le puede empezar a agregar los detalles que desea. Cada casa tiene características básicas como: metros cuadrados, número de plantas, si es esquinera o no, orientación (norte, sur, este u oeste), si tiene patio grande o pequeño, numero de habitaciones, número de baños. Pero se puede elegir ciertos acabados y elementos adicionales como: pisos de porcelanato nacional o importado, grifería estándar o italiana, iluminación tradicional o led, baños insonorizados, aislante térmico en el techo. Por cada uno de estos detalles se cobra un valor adicional.
7. Existen 3 casas básicas: casa Cielo, casa Paraíso y casa Oasis. Entre ellas la casa Oasis es la más pequeña, con solo 2 habitaciones, mientras la casa Paraíso tiene 3 y la Cielo tiene 4. (Por ahora usted decida el resto de las características).
8. El cliente puede diseñar su casa, pero sólo podrá ver el costo final o enviárselo por email (PDF) siempre y cuando esté registrado, caso contrario el sistema debe pedirle que para completar dichas operaciones debe registrarse y se le enviarán automáticamente el finalizar el registro. Además, si es un cliente registrado o un vendedor, también puede ver el precio de cada característica desde el momento en que la vaya a elegir, en cambio esto no lo podrían ver los clientes no registrados.

Identificar las características que marcan el perfil de quienes están dispuestos a pagar el impuesto.

Quienes están dispuestos a pagar el ICE alas fundas plásticas

# Propuesta

## C:\Users\Tania\Documents\yuly\diseño de software\proyecto2DS\proyecto2DS_CasosUso.pngDiagrama de Casos de Uso

Ilustración 1. Diagrama de Casos de Uso

Tabla 1. Especificación de Caso de Uso "Diseñar Casa".

|  |  |
| --- | --- |
| **Caso de Uso** | |
| **ID** | **001** |
| **Título** | Diseñar Casa |
| **Autor** | Yuleixi Gracia |
| **Actor Principal** | Cliente registrado (puede ser un cliente o un vendedor) |
| **Actor Secundario** |  |
| **Usa** |  |
| **Extiende** | Elegir opción |
| **Descripción** | **Cliente:** Diseña una casa basándose en alguno de los modelos básicos de la empresa. |
| **Precondición** | El cliente debe de iniciar sesión, elegir una de las casas básicas y elegir la opción de diseñar casa. |
| **Secuencia Normal (Éxito)** | 1. El cliente inicia sesión.  2. El cliente elije una de las casas básicas.  3. El cliente elije la opción de diseñar casa  4. El cliente diseña una casa a su gusto. |
| **Pos-condición** | Agregar detalles  Elegir acabados  Elegir elementos adicionales  Ver costo final de la casa  Enviar por correo el costo final de la casa |
| **Secuencia alternativa** | 1a. No puede iniciar sesión  1. El cliente no puede ingresar al sistema  2. El cliente no está en la base de datos.  3. El cliente debe ser registrado por un empleado.  4. El cliente vuelve al caso de uso diseñar casa. |
|  | 3a. El cliente no elige una casa básica.  1. El cliente no puede diseñar una casa.  2. El cliente debe elegir una casa básica.  3. El cliente vuelve al caso de uso diseñar casa. |

|  |  |
| --- | --- |
| **Caso de Uso** | |
| **ID** | **003** |
| **Título** | Realizar Diseño de Casa |
| **Autor** | Yuleixi Gracia |
| **Actor Principal** | Cliente No registrado |
| **Actor Secundario** |  |
| **Usa** |  |
| **Extiende** | Elegir opción |
| **Descripción** | **Cliente:** Agrega acabados a una casa base. |
| **Precondición** | El cliente debe elegir una de las casas básicas, elegir la opción de diseñar casa y agregarle acabados. |
| **Secuencia Normal (Éxito)** | 1. El cliente elije una de las casas básicas.  2. El cliente elije la opción de diseñar casa.  3. El cliente agrega detalles.  4. El cliente le agrega los acabados deseados.  5. El cliente elije elementos adicionales. |
| **Pos-condición** | Ver costo final de la casa  Enviar por correo el costo final de la casa  Registrarse |
| **Secuencia alternativa** | 1a. El cliente no elige una casa básica.  1. El cliente no puede diseñar una casa.  2. El cliente debe elegir una casa básica.  3. El cliente vuelve al caso de uso diseñar casa. |

Tabla 2.Especificaion de Caso de Uso "Realizar Diseño de Casa".

## Patrones de Diseño

### Builder:

Como Patrón de diseño, el patrón builder (Constructor) es usado para permitir la creación de una variedad de objetos complejos desde un objeto fuente (Producto), el objeto fuente se compone de una variedad de partes que contribuyen individualmente a la creación de cada objeto complejo, en nuestro caso, se usara para poder tener varios tipos de casa usando el mismo código de construcción y también para que no se afecte el resto del programa al momento de querer agregar otro tipo de casa.

### Decorator:

El patrón Decorator responde a la necesidad de añadir dinámicamente funcionalidad a un Objeto ,este patrón de diseño facilitará la decoración de las casas y aunque se agregue un nuevo acabado solo será otra extensión de la interfaz casa sin generar acoplamiento entre las demás clases.

### Singleton:

En ingeniería de software, singleton o instancia única es un patrón de diseño que permite restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Para poder conectar la base de datos se necesita una clase que solo se encargue de eso de modo que los usuarios no accedan a la base de datos de forma directa ni se creen varias instancias de esa clase, puesto que la conexión del programa siempre es a la misma base de datos y a la misma instancia de conexión.

## C:\Users\Tania\Documents\yuly\diseño de software\proyecto2DS\proyecto2DS_DiagramaClases.pngDiagrama de clase

Ilustración 2. Diagrama de Clases.

## Diagramas de Secuencia

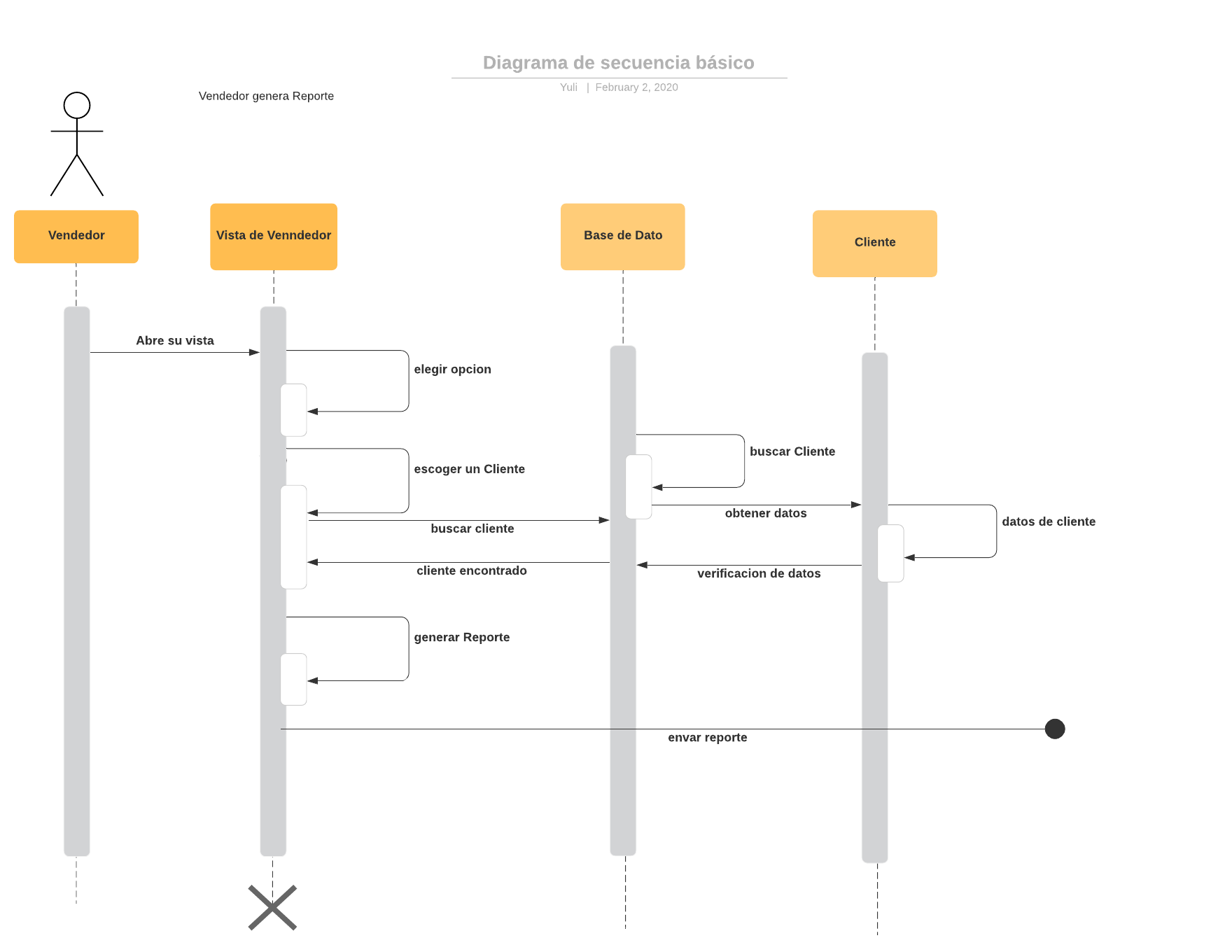


Ilustración 3. Vendedor Genera Reporte

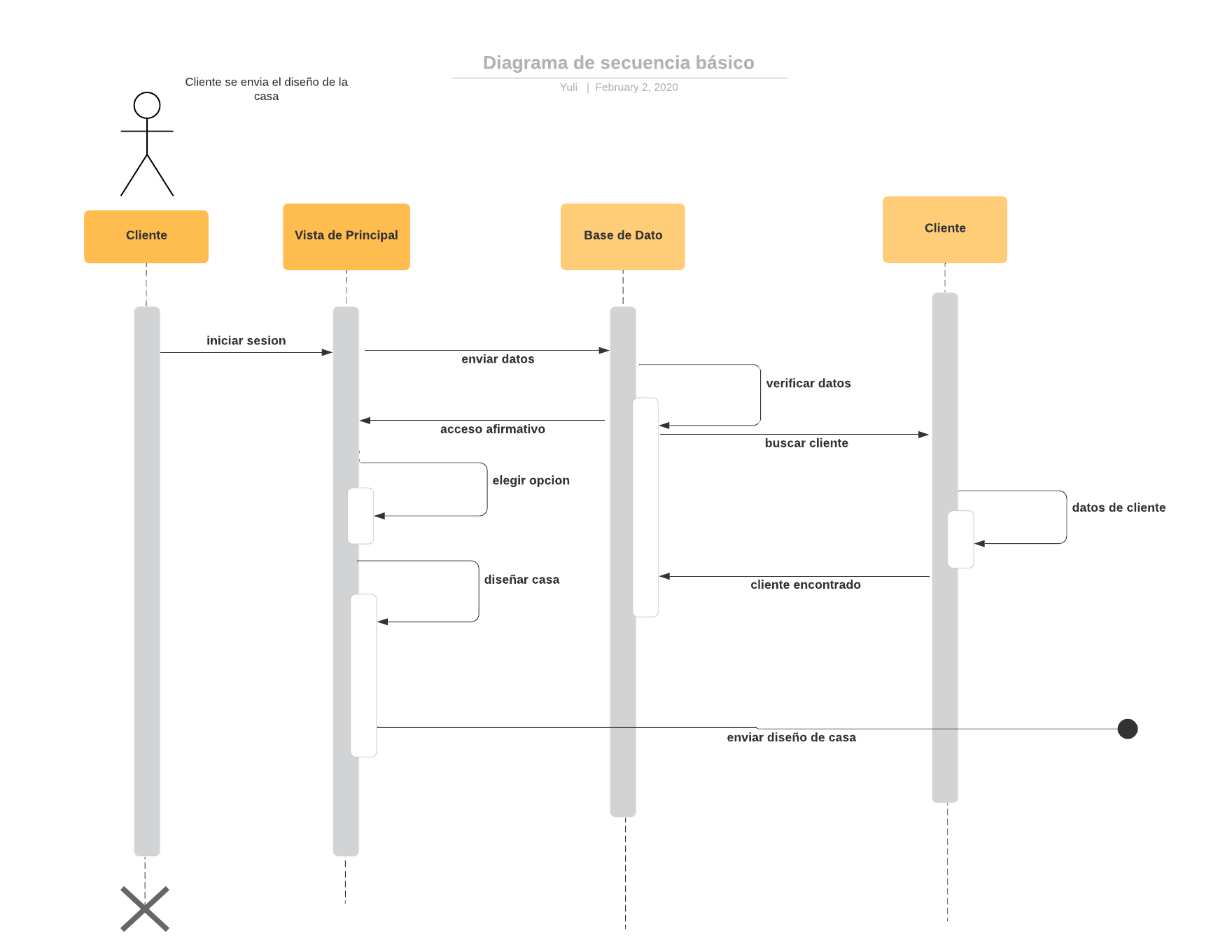
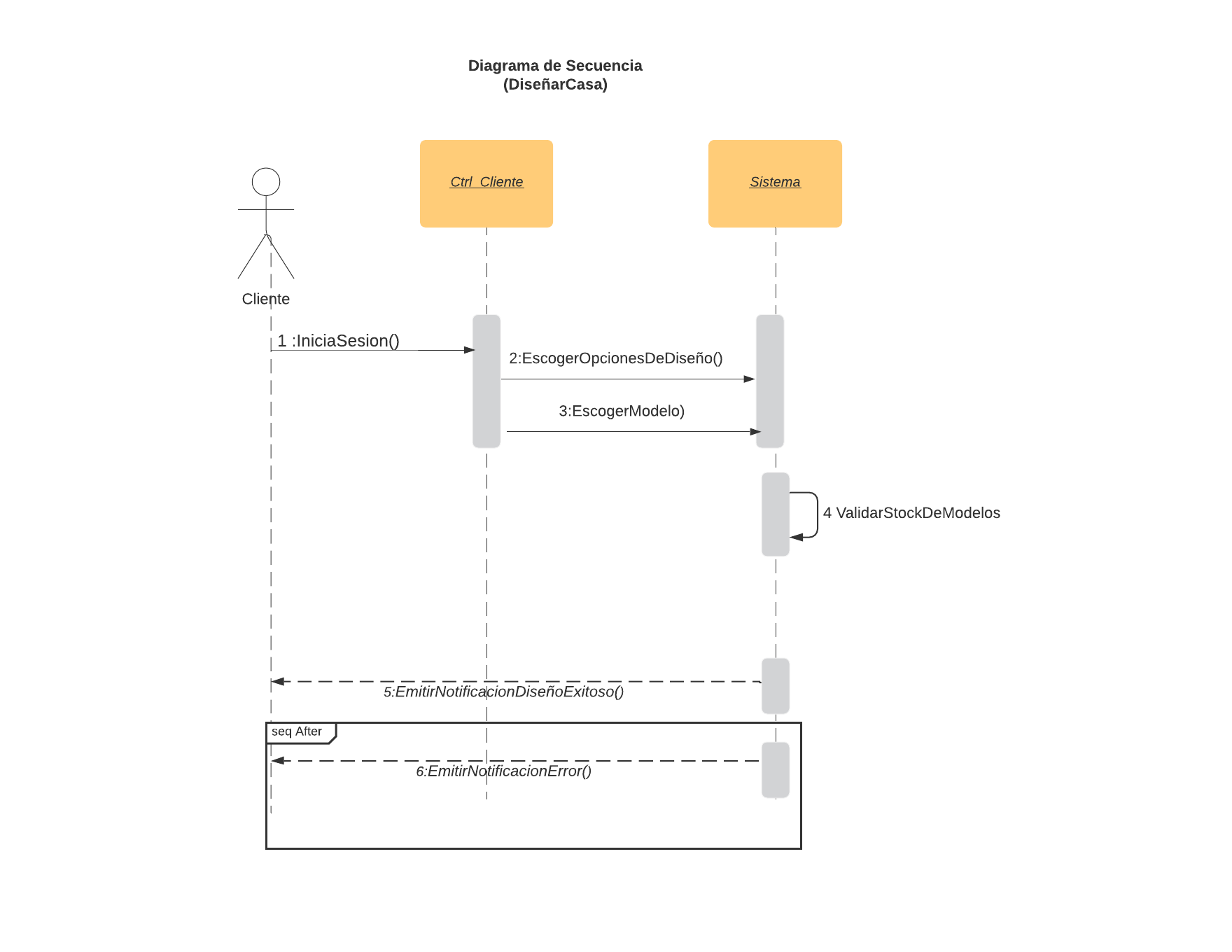
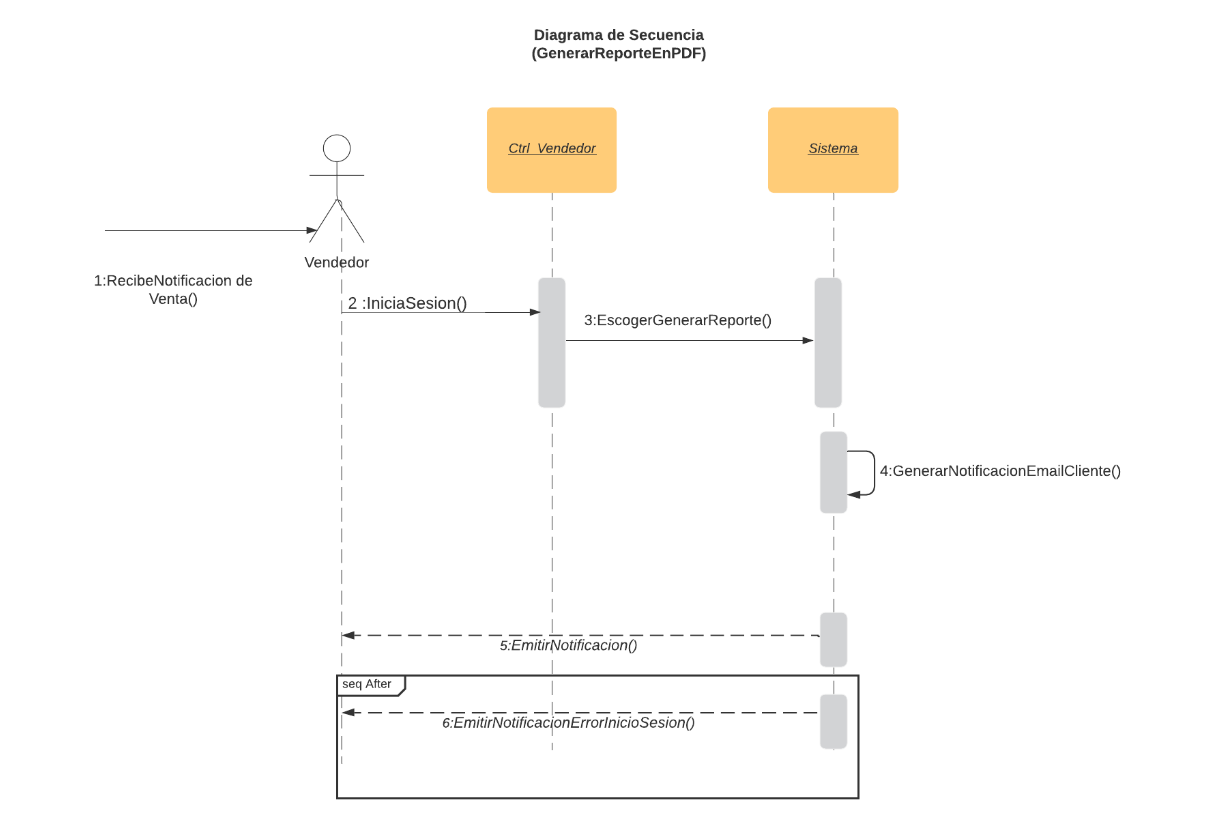
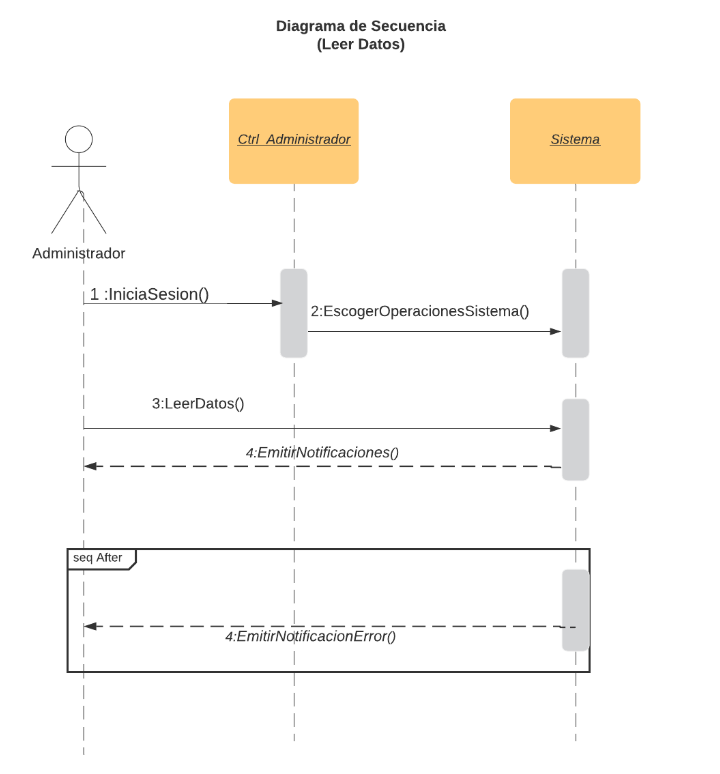
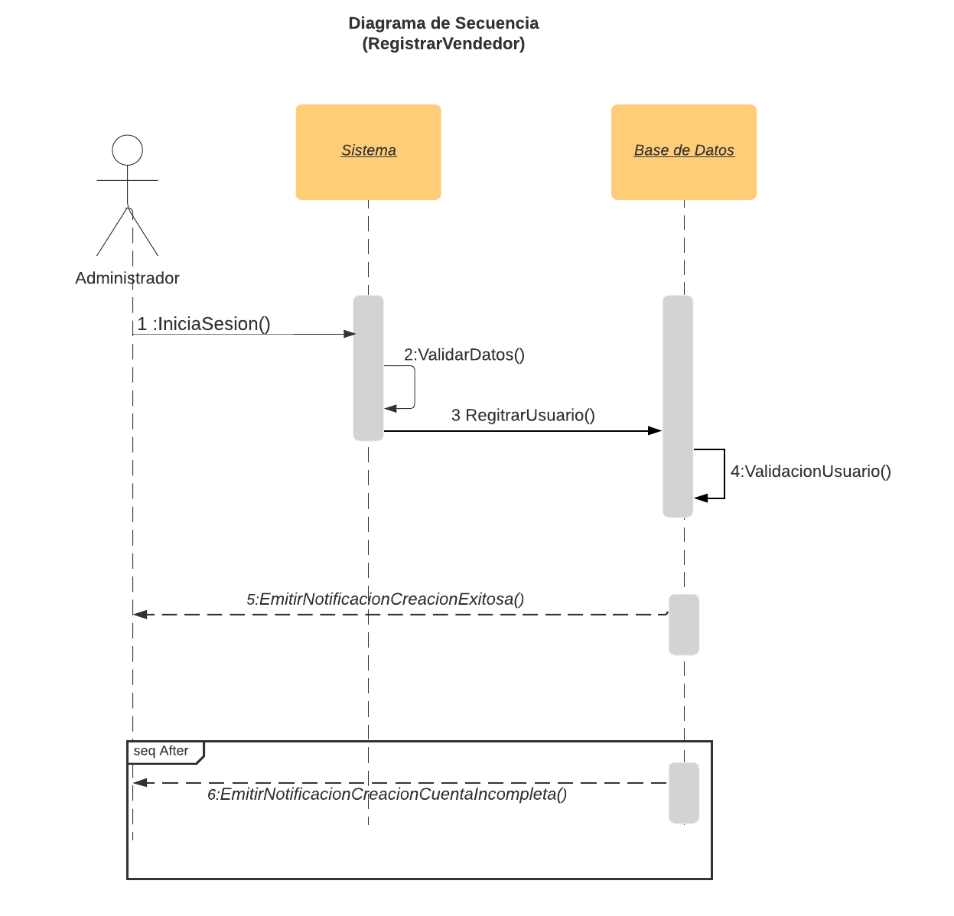


Ilustración 4. Cliente envía diseño de casa









## C:\Users\Tania\Documents\yuly\diseño de software\proyecto2DS\proyecto2DS_baseDatos.pngDiagrama Lógico de la Base de Datos

Ilustración 5. Diagrama Lógico de la Base de Datos.

# Prototipo

## Docker

Requerimientos

- Tener instalado docker-compose (Teniendo instalado Docker Desktop ya se instala autom\'e1ticamente docker-compose)\

Pasos

1.Desde la terminal ir a la carpeta dentro del proyecto llamada dockerFiles mediante el comando cd

2. Ejecutar el comando docker-compose up con el cual se instala la imagen de mysql en caso de no tenerla instalada y una vez instalada se crea el contenedor y empieza a ejecutarse.\

3. Cambiarse a Linux o Windows segun en el sistema que hayamos trabajado el archivo docker.

4. Conectarse a la base de datos con el usuario root y contraseña proyecto. La direccion y puerto de enlace es localhost:3309

## SonarQube

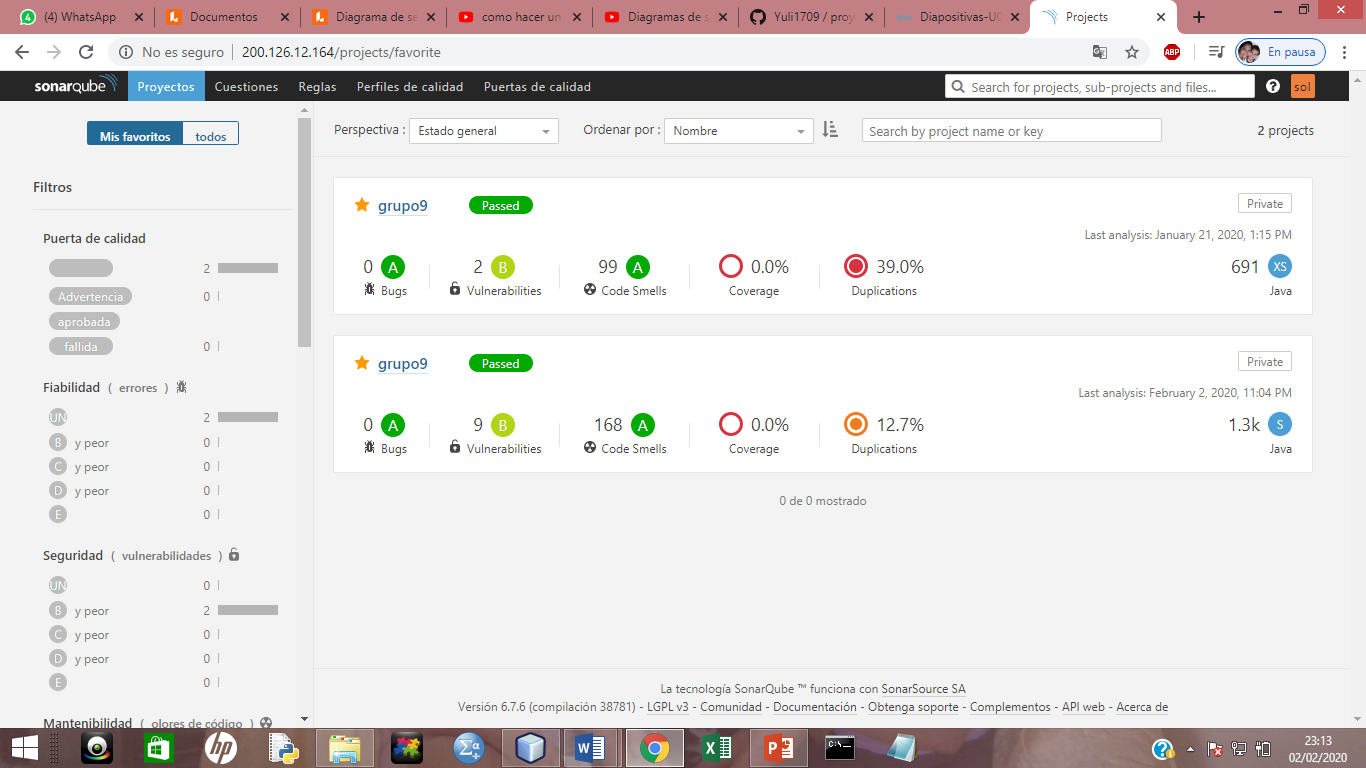


Ilustración 6. Reporte SonarQube.

## Unit Test

## Este test nos permitirá validar que cuando se esté creando una conexión está en el caso de ser nula crea una conexión y posteriormente la retorne por lo que al el condicional de que la conexión una vez instanciada será nula efectivamente debe ser falso y por lo tanto el assertFalse nos permitirá conocer que está funcionando correctamente.

Verifica que los valores instanciados por default del método construircasaEspecial de la clase cielo\_builder estén siendo ingresados correctamente al crear una instancia de casa con los valores esperados en este caso del precio de una casa .

Este test nos verifica que al usar los 2 constructores de la clase usuario y le pasamos los mismos atributos ( usuario y contraseña) deben ser los mismos objetos por lo que lo verificamos con un assertEquals de los 2 objetos instanciados.

# Referencias

* Reporte SonarQube: <http://200.126.12.164/dashboard?id=proyDS9%3Agrupo9>

Blankenberg, D., Kuster, G. V., Coraor, N., Ananda, G., Lazarus, R., Mangan, M., ... & Taylor, J. (2010). Galaxy: a web‐based genome analysis tool for experimentalists. Current protocols in molecular biology, 19-10.

Bolger, A., & Giorgi, F. Trimmomatic: A Flexible Read Trimming Tool for Illumina NGS Data. URL http://www. usadellab. org/cms/index. php.