## Entregable # 2 – Arquitectura MVC + Servicios + Docker

| | Nombre |
|---|---|
| **Equipo de Trabajo** | Rafael VIllegas Michel |
| | Santiago Soto Marulanda |
| | Felipe Cortés Jaramillo |

**Nombre proyecto:** Mr. Watch

### 1. Modelo verbal definitivo

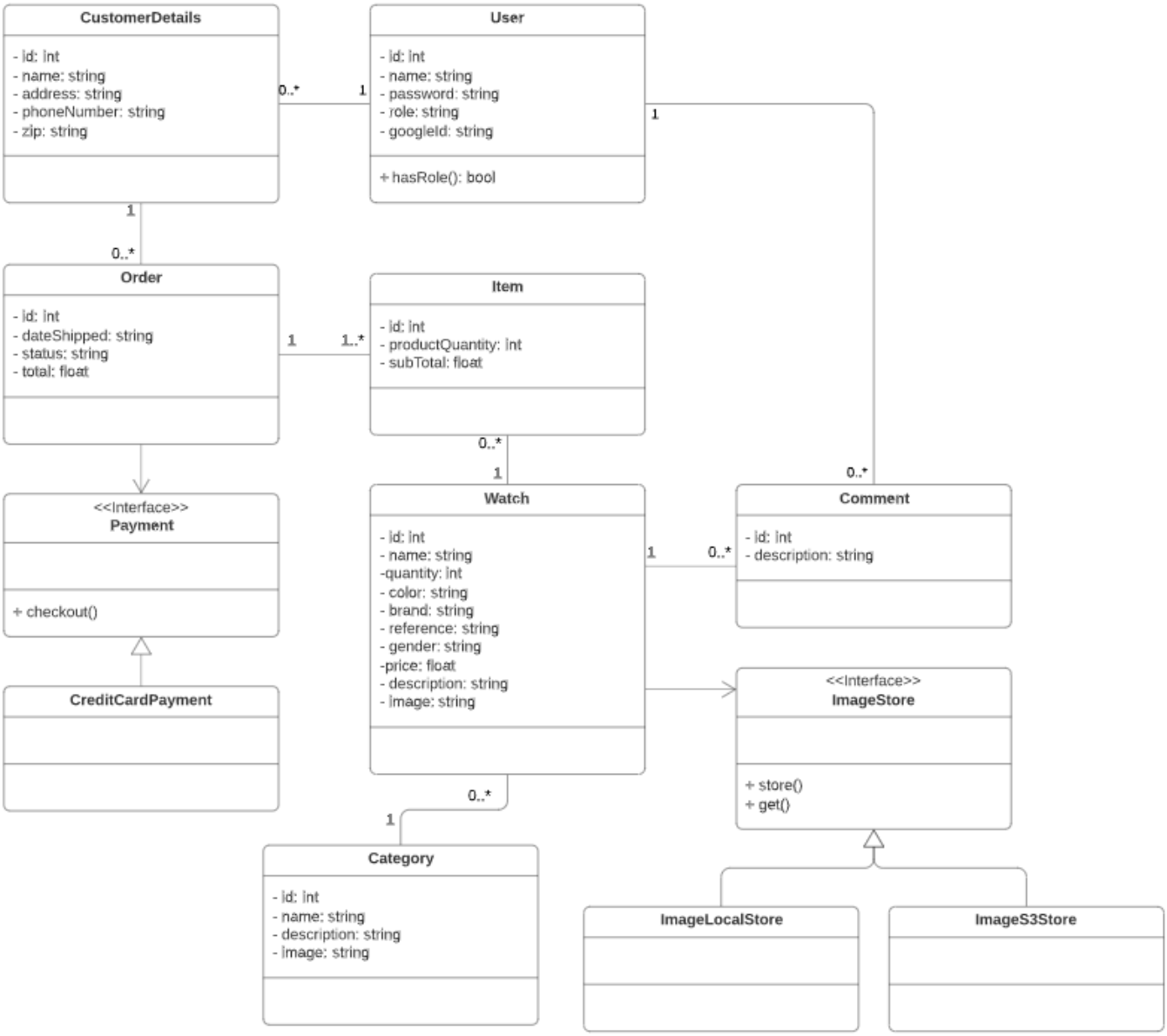
El proyecto consiste en la elaboración de una página web de una comercializadora de relojes de todo tipo. En ella, los usuarios administradores podrán hacer funciones básicas de CRUD sobre su inventario de relojes, mientras que los compradores podrán visualizar todos los productos de la tienda y comprar el que se les antoje. Pueden observar todo tipo de relojes y dejar un comentario en cada uno de ellos. Asimismo, los usuarios tendrán la posibilidad de revisar productos de una tienda aliada la cual ofrece todo tipo de productos.

Para esta entrega, el alcance de este proyecto refleja un mínimo producto viable, en el que el admin tiene funcionalidades de CRUD sobre los relojes y categorías, además de poder descargar la lista de relojes en stock en formato CSV. Por otro lado, el usuario podrá ver los relojes por categorías, por más vendidos y ordenados por nombre, relevancia y precio, para elegir cuál añadir a su carrito de compras y luego proceder con la transacción.

Por otro lado, se busca que la aplicación contenga como mínimo dos inversiones de dependencias, las cuales se aplicaron en el almacenamiento de las imágenes y en el método de pago. Se busca un despliegue por Docker y que consuma y exponga una Api.

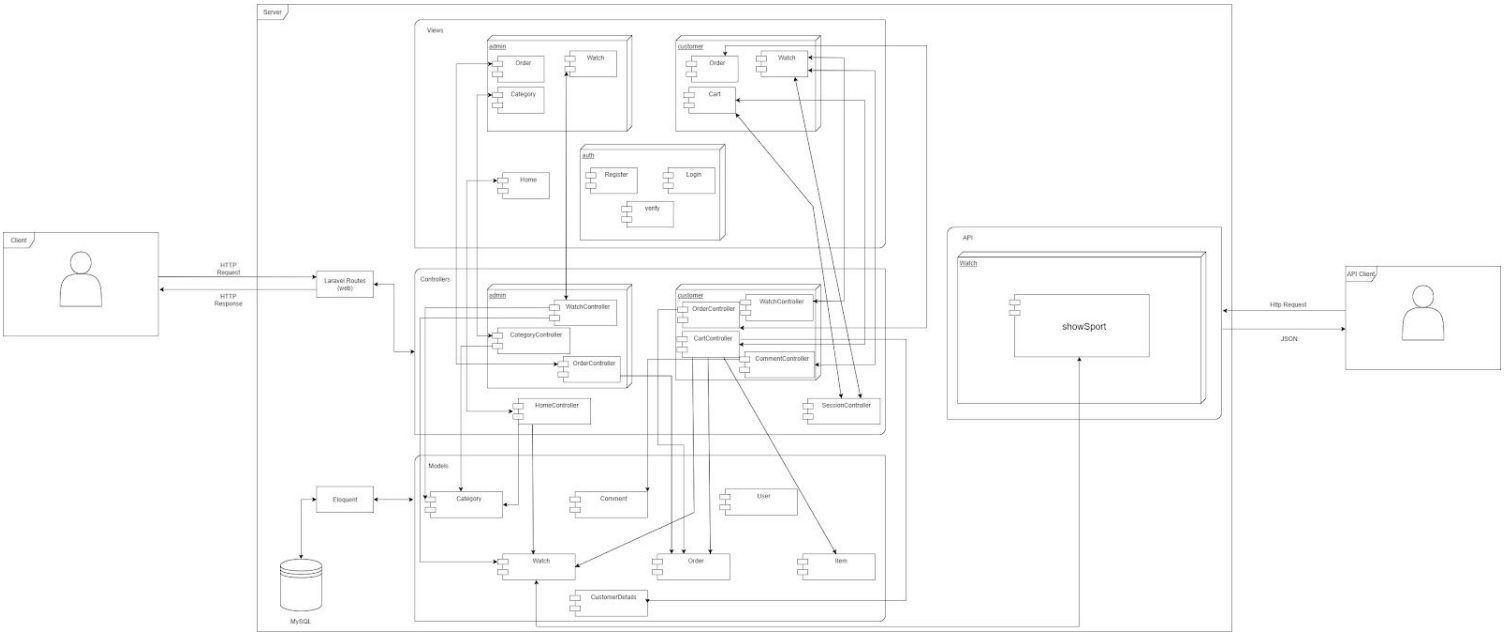
El beneficio de esta aplicación está en el manejo cuasi automático del inventario de relojes, así como en la facilidad de tener una recopilación automática de los datos de los compradores, lo cual permitiría hacer distintos análisis exploratorios o de mercado en un futuro, si la empresa de relojes así lo desea.

## 2. Diagrama de clases




https://www.lucidchart.com/invitations/accept/200514a3-e68f-4060-a457-b3dea5b03bd1

## 3. Diagrama de arquitectura




https://drive.google.com/file/d/1WmQkWw8JyXrlm8To04o9tWfN3pHF1-gV/view?usp=sharing

## 4. Implementación en Laravel

**Instrucciones para el arquitecto:**

- Cree un repositorio en GitHub (rama master).
- Clone el repositorio localmente, luego cree un proyecto en laravel y súbalo al repositorio GitHub.
- Comparta el repositorio con sus compañeros.
- Repártales a los compañeros las clases que deberán programar. Por ejemplo, si su proyecto consta de 6 clases, repártales de a 2 o 3 clases a cada compañero. Usted como arquitecto tiene total libertad de la repartición de las clases. Recuerde que usted deberá sacar tiempo para la elaboración de unos documentos y para revisar cada código nuevo que intenten aplicar sus compañeros.
- Sus compañeros nunca podrán hacer “push” directo a la rama master.
- Cuando un compañero decida trabajar en el proyecto, deberá: (i) crear una nueva rama, o (ii) realizar un fork del proyecto. Y cuando un compañero desee aplicar los cambios a la rama principal (master o development) deberá realizar un pull request desde GitHub.
- El arquitecto analizará el pull request y solo aceptará los cambios si los considera pertinentes.
- Defina en GitHub un directorio de documentos de codificación. Ese directorio incluirá 2 archivos:

    - **Guía de estilo de programación.** Defina un documento corto (de no más de 2 páginas) con todas las indicaciones de estilo de codificación. Por ejemplo, defina estilo para los ifs, para los while, como nombrar las clases, los métodos, las variables, etc. Ejemplos: https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-1-basic-coding-standard.md - https://guidelines.spatie.be/code-style/laravel-php
    - **Reglas de programación.** Defina otro documento corto (de no más de 2 páginas) con las reglas que usted considera esenciales de programación en su proyecto Laravel. Divida las reglas por categorías (reglas para controladores, para modelos, para vistas, para rutas, etc). Ejemplo de reglas esenciales: (i) nunca haga un echo en un controlador. (ii) Toda ruta debe estar asociada a un controlador. (iii) Toda vista debe extender del layout master. (iv) Todas las vistas deben ser Blade. (v) nunca abra y cierre php dentro de las vistas. Etc.
    - Si un compañero envía un pull request que no siga las reglas definidas en los dos documentos anteriores, remítalo a esa documentación.
- Cree un tablero en asana, trello, o cualquier otro sistema donde pueda controlar las actividades y tareas pendientes del equipo.

**Instrucciones globales:**

- El sistema debe ser un proyecto web con Laravel y base de datos MySQL.
- Deberán implementar todas las clases de su proyecto.
- Deberán utilizar el sistema “migrations” de Laravel para llevar el registro de cambios del SQL.
- Utilice fakers para facilitar la ejecución inicial del programa, o como plan b, provea un SQL con inserción de datos ficticios.
- Recuerde que cada línea en el diagrama de clases se convierte en funciones (dinámicamente propiedades) en las dos clases que se relacionan.
- Cree un archivo README.TXT donde explique cómo ejecutar el programa, cual es el archivo principal que se debe invocar, entre otros.
- Utilice el sistema de Login que ofrece Laravel (todo un sistema de Login se puede crear en menos de 15 minutos). Ese sistema esta asociado a la clase user que ya viene con los proyectos de Laravel.
- La aplicación deberá contener por lo menos “7 funcionalidades interesantes” diferentes a las tradicionales: crear, editar, borrar y leer. Ejemplo: (i) búsqueda de productos, (ii) ver top 3 productos más vendidos, (iii) generar en pdf la factura de venta, (iv) descargar mis notas en excel, etc.
- **Detalle las funcionalidades principales de su aplicación:** haga una tabla en excel, con 3 columnas. Columna 1 nombre de la funcionalidad, Columna

2 nombre del archivo donde está implementada la funcionalidad, Columna 3 línea desde la cual inicia la implementación de la funcionalidad. Agregue el archivo a GitHub. Liste las funcionalidades interesantes al final.

- Tome un pantallazo de las 3 secciones más importantes de la aplicación guárdelas en GitHub.
- **Sugerencia 1:** para facilidad de este proyecto, olvídese de las herencias de clase usuario, y "junte" todas clases hijas de usuario, en usuario. Agréguele un atributo tipo para identificar el tipo de usuario.
- **Sugerencia 2:** si ejecuta la "sugerencia 1" y el diagrama le queda con pocas clases (4 o menos), cree nuevas clases que le puedan brindar funcionalidades interesantes al proyecto. Ejemplo: clase comentario, clase puntuación, clase tema/post, clase histórico, clase favoritos, clase mi lista, etc.
- Todos los textos del proyecto deben ir resources/lang/* - si le da tiempo, implemente el proyecto en 2 idiomas.
- Aplique principios DRY y ETC. Sugerencia: ver libro "Pragmatic programmer" topic 8 y 9 (son solo 4 páginas).
- Aplique los principios, tips, y sugerencias que se han brindado durante todo el curso. Muchos de ellos no aparecen en las presentaciones ya que se discutieron en clase.
- Ojo al reutilizar el código de los talleres, aunque es una muy buena base para iniciar, hay varios cambios y mejoras que se deben realizar que se discutieron durante las clases.
- Implemente 2 pruebas unitarias. 4

**Instrucciones de entrega para el arquitecto:**

- Suba el proyecto a la cuenta que se le brindó en AWS.
- Cree un .zip de todo el proyecto y súbalo por interactiva virtual (en caso de que no se pueda por cuestiones de tamaño, envíelo por wetransfer o mega al correo del docente).
- Suba este documento al repositorio de GitHub. Y finalmente comparta el link del repositorio con el docente antes de la fecha de finalización de entrega.

**Sugerencia:**

Aproveche el espacio del 3 de marzo que se dará para resolver dudas del parcial y del entregable. Se espera que para esa clase todos los equipos ya tengan implementado el 80% o más de su respectivo proyecto.

**Instrucciones para el arquitecto de usabilidad (no puede asignársele esa responsabilidad al arquitecto del proyecto pasado):**

· Como arquitecto de usabilidad usted será responsable por que la aplicación sea lo más usable posible. Deberá verificar que todo el sistema desarrollado cumpla con buenos estándares de usabilidad, y deberá asignar a los desarrolladores correspondientes los cambios que considere pertinentes.

· Aspectos para tener en cuenta como arquitecto de usabilidad:

o Que todas las vistas de la aplicación manejen la misma estructura visual. Que se utilicen los mismos colores, fondos, tipos de letra, etc.

o Que los formularios estén bien diseñados y contengan una estructura coherente.

o Que los formularios no se vacíen (y toque volverlos a llenar) si se encuentran errores.

o Que los campos de los formularios estén bien diseñados (campos de textos, campos de selección, radio buttons, textarea, etc).

o Que la aplicación tenga botones de fácil acceso (menú principal, menús laterales, pie de página correspondiente, etc).

o Que la aplicación cuenta con un sistema de navegación (buscar en Google "breadcumbs").

o PLUS: que la aplicación se vea bien, tanto en computador como en celular.

· Cree un tablero en asana, trello, o cualquier otro sistema donde pueda controlar las actividades y tareas pendientes del equipo en cuanto a pendientes de usabilidad. · Lectura recomendada: https://www.quicksprout.com/website-usability/

**Instrucciones globales (entregable #2):**

· Implemente todo lo que se pidió para el entregable #1 (si es que aún no lo ha hecho / ver instrucciones anteriores).

· Implemente las 7 funcionalidades interesantes de la entrega anterior (si es que aún no lo ha hecho).

· Haga todas las correcciones que el docente le notificó tanto en las sustentaciones pasadas, como en el documento entregado a cada equipo. Si no se realizan las correcciones a la entrega pasada, el docente no seguirá calificando esta entrega y se colocará nota entre 0 y 1.

· Separe las operaciones administrativas de la aplicación, de lo que ven los usuarios (cree un panel de administración -> si es que aún no lo ha hecho).

· En esta entrega el sistema se deberá implementar en 2 idiomas (recuerde, nada de textos “quemados” ni en controladores, ni en vistas, siempre use LANG).

· Implemente un servicio web donde usted provea en formato JSON, información relevante de su aplicación.

o Por ejemplo: (i) lista de productos en stock en venta, incluyendo el enlace directo a la visualización de cada producto; (ii) lista de estudiantes con su nota acumulada; (iii) lista de rutinas de gimnasio, (iv) listado de gafas en promoción, (v) lista de mascotas para adoptar, etc.

o El equipo siguiente deberá consumir ese servicio, y mostrarlo en la aplicación de ese equipo.

o Por ejemplo: El equipo 1 provee un servicio donde despliega información de sus productos. El equipo 2 consume ese servicio, crea un nuevo controlador y vista, y crea una ruta: /productos-aliados y ahí despliega la información que consumió del servicio del Equipo 1. Adicionalmente, el equipo 2 debe proveer un servicio, que deberá ser consumido por el equipo 3, etc.

· Dentro de su aplicación, consuma el servicio del equipo anterior (ver explicación anterior).

· Dentro de su aplicación, consuma un servicio de una compañía tercera (ya sea Google, Facebook, un API en la web, etc). Por ejemplo, el equipo 3,

podría consumir un servicio para mostrar el clima actual en Medellín, y desplegarlo en la parte superior de la cabecera de la aplicación.

· Implemente 2 inversiones de dependencias. Sugerencia: implemente inversión de dependencias para guardar imágenes de manera local y en S3. E implemente otra inversión de dependencias para simular pagos con cheques, o pagos donde le reste dinero a la cuenta del usuario (simulando que el usuario tiene un atributo con su dinero disponible).

· Despliegue la aplicación con Docker (en AWS).

· Implemente un sistema de nombre de dominio simplificado (que no toque acceder por la IP de Amazon) -> utilice dominios gratis .tk http://www.dot.tk/.

· Implemente mejoras que considere pertinentes, paginación, carga de archivos, mejora en los sistemas de fakers, menús laterales, un sistema de banner en la página principal, entre otros.

**Instrucciones de entrega para el arquitecto:**

· Suba el proyecto a la cuenta que se le brindó en AWS (despliéguelo con Docker).

· Cree un .zip de todo el proyecto y súbalo por Teams (en caso de que no se pueda por cuestiones de tamaño, envíelo por wetransfer o mega al correo del docente).

· Suba este documento al repositorio de GitHub. Y finalmente comparta tanto el link del repositorio, como el dominio .tk con el docente antes de la fecha de finalización de entrega