Instituto Tecnológico de Costa Rica

Proyecto Fase 1 - Parte 1

Software Architecture Document

Version <1.3>

Revision History

Date	Version	Description	Author
16/04/2021	1.0	Inicio del documento	Eduardo
17/04/2021	1.1	Inicio de rellenar la introducción	Agustín
18/04/2021	1.2	Relleno de campos Arquitectura de Información	Agustín
19/04/2021	1.3	Agregar campos restantes	Agustín

Table of Contents

Software Architecture Document

1. Introducción

1.1 Propósito

El propósito de este proyecto es construir una aplicación web para solucionar un problema de la vida real. Este problema se refiere al de un cliente, el cual es un grupo de emprendedores que son dueños de un centro de entrenamiento o gimnasio, que requiere una página para mantener el control de los servicios que se brindan. Esto dado que gracias a la pandemia del COVID-19, muchos negocios de prestación de servicios han sufrido una pérdida de ingresos bastante significativa. Esto hace el llamado a la administración por medios digitales de muchos negocios que previamente se manejaban de manera presencial.

1.2 Ámbito

Este documento de arquitectura de software provee una visión arquitectónica del sistema del centro de entrenamiento, el cual será desarrollado con el Web Stack MERN (MongoDB, Express, React, Node.js). Dicha aplicación será desarrollada siguiendo el patrón de diseño de software MVC, correspondiente a las siglas de “Model - View - Controller”, y utilizando programación interna sobre un paradigma orientado a objetos. En dicha arquitectura se propone que las capas de Modelo y Controladores serán parte del *back-end* y la capa de Vista será el *front-end*.

1.3 Definiciones, Acrónimos y Abreviaciones

- MVC: Siglas para Model - View - Controller
- Web Stack: Colección de software requerido para el desarrollo de páginas web.
- MERN: Siglas para MongoDB-Express-React-Node.js
- Back-end: Se refiere a la capa del software que maneja controla las llamadas lógicas que provienen del Front-end.
- Front-end: Se refiere a la capa del software con la que el usuario interactúa, la interfaz de usuario.
- UML: Siglas para Unified Modeling Language

1.4 Referencias

[1] MongoDB. (2021, abril). *Cluster-Autoscaling*. Consultado en https://docs.atlas.mongodb.com/cluster-autoscaling/

2. Representación de Arquitectura de Información

El sistema a ser desarrollado tendrá una arquitectura de 4+1 vistas según Philippe Kruchten. Dichas vistas son la Vista Lógica, Vista de Procesos, Vista de Desarrollo, Vista Física y la Sección de Escenarios o Casos de Uso. Para cada una de las vistas se necesitan ciertas representaciones de forma diagramada, las cuales son respectivamente:

- Vista Lógica: Diagrama de Clases y de Secuencias
- Vista de Procesos: Diagramas de Actividad
- Vista de Desarrollo o Despliegue: Diagrama de Componentes y Paquetes
- Vista Física: Diagrama de Despliegue
- Sección de Escenarios o Casos de Uso: Diagramas de Casos de Uso

Para cada uno de estos diagramas, se utilizará el estándar de UML.

3. Objetivos y Limitaciones

Algunos de los objetivos del sistema, o requerimientos no funcionales como se conocen, son que este cumpla con tener:

- Seguridad
- Alta disponibilidad
- Usabilidad
- Eficiencia
- Mantenibilidad

Dicho sistema debe ser construido bajo ciertas condiciones, entre las cuales incluyen:

- Utilizar programación interna bajo un paradigma orientado a objetos.
- Separar los datos y la lógica de negocio bajo la arquitectura MVC.
- Utilizar algún patrón de diseño creacional, el cual será para este caso el Factory

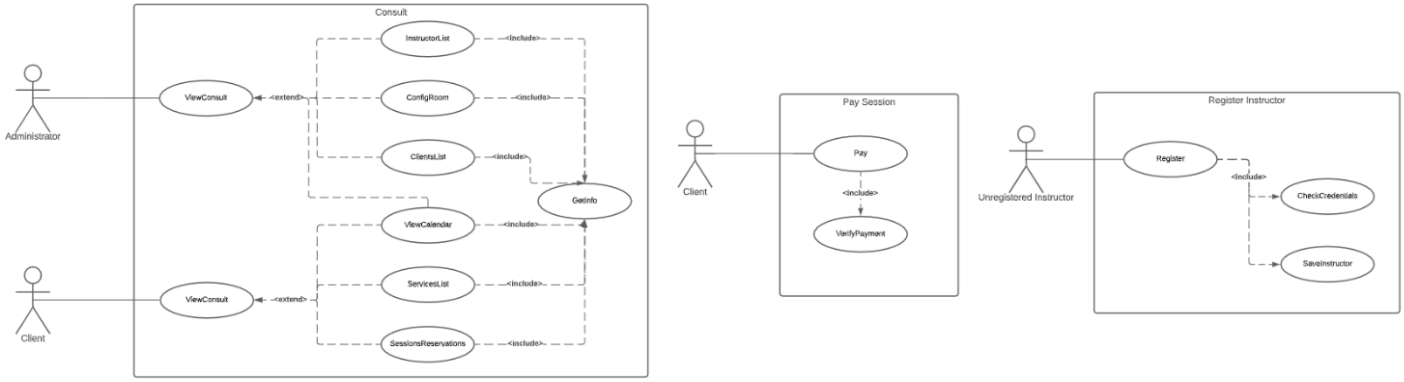
Method.

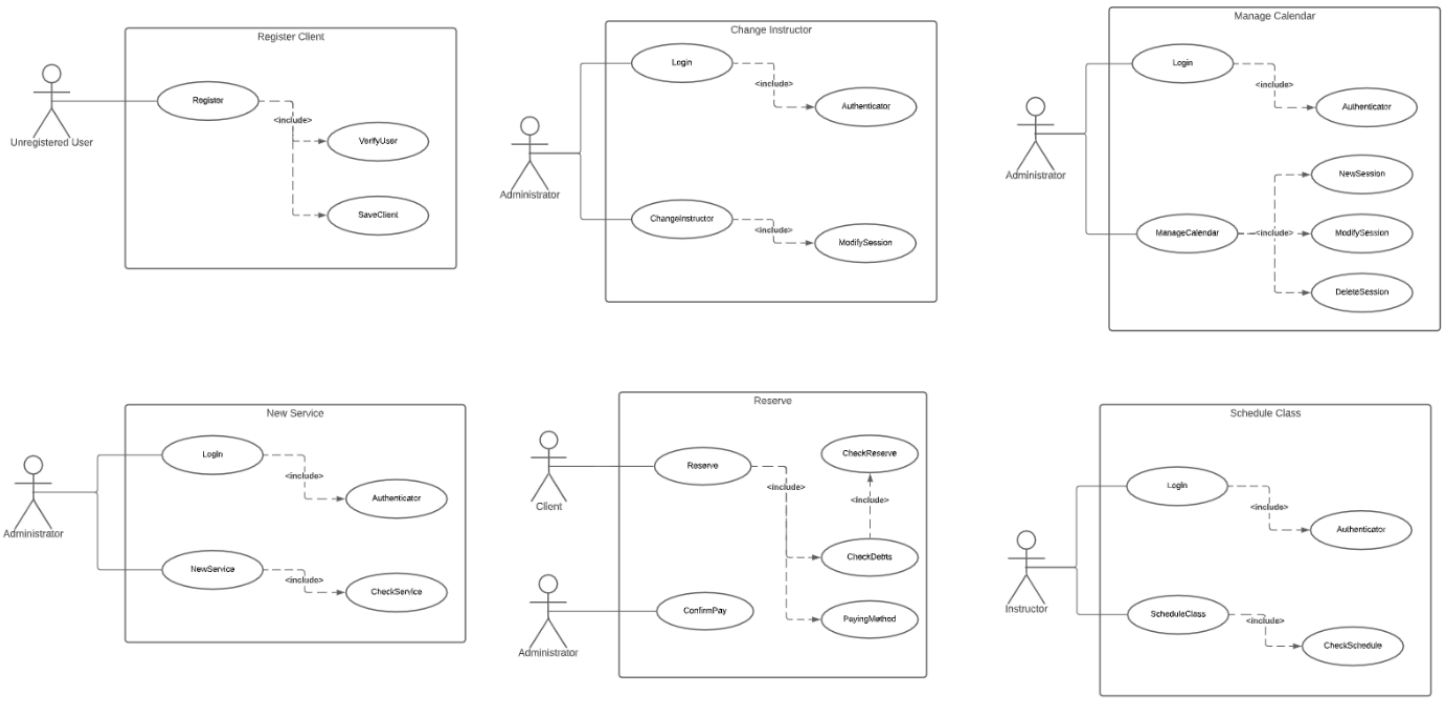
- Desarrollar la aplicación utilizando el Web Stack MERN.

4. Casos de Uso

La Sección de casos de uso está adjunta abajo con los diagramas de casos de uso, y también se encuentra disponible para mejor visualización en el siguiente enlace:

https://github.com/agusbrenes/Proyecto-DIS-Gimnasio/blob/main/Diagramas/Casos%20de%20Uso.pdf





5. Vista Lógica

La Vista Lógica con los diagramas de clases y de secuencias se encuentra disponible para mejor visualización en los siguientes enlaces:

Diagrama de Clases:

https://github.com/agusbrenes/Proyecto-DIS-Gimnasio/blob/main/Diagramas/Vista%20L%C3%B3gica%20Clases.pdf

Diagramas de Secuencias:

https://github.com/agusbrenes/Proyecto-DIS-Gimnasio/blob/main/Diagramas/Vista%20L%C3%B3gica%20Secuencias.pdf

5.1 Paquetes

5.1.1 Model

- Capa de lógica de negocio y tipos de datos utilizados en el sistema.

5.1.2 Controller

- Capa de control de datos y procesos de la aplicación.

5.1.3 View

- Capa de interfaz de usuario gráfica, ventanas de la página web donde se desarrollará.

5.2 Algunos Casos de Uso

Nombre	Consulta
Autor	Jesús Andrés Chavarría Delgado
Fecha	17/04/21
Prioridad	N

Descripción:
Este caso de uso permite a los usuarios (ya sean clientes o administradores) realizar las consultas que necesiten de acuerdo a sus necesidades.

Actores:
Cliente registrado y Administrador

Precondiciones:
El cliente o el administrador deben estar registrados en la base de datos y conectados al sistema

Flujo Normal:
1. El Cliente/Administrador abre la ventana de consultas.
2. Se selecciona la consulta a realizar (estas dependen si es Cliente o Administrador).
3. Se confirma la selección.
4. Se despliega la consulta en la pantalla.

Flujo Alternativo:
1. El Cliente/Administrador abre la ventana de consultas.
2. Se selecciona la consulta a realizar.
3. Se confirma la selección.
4. Un error de conexión ha ocurrido.
5. Se despliega un mensaje de error en la pantalla.

Poscondiciones:
La consulta queda desplegada en la pantalla hasta que el usuario decida.

Nombre	Registrar Cliente
Autor	Jesús Andrés Chavarría Delgado
Fecha	17/04/21
Prioridad	N
Descripción: Permite Registrar un nuevo cliente en la base de datos	
Actores: Cliente no registrado	
Precondiciones: El cliente no debe estar registrado en la base de datos	
Flujo Normal: 1. Cliente abre la ventana de registrar 2. Ingresa los datos y presiona registrar 3. Se verifican los datos en la base de datos 4. Se guarda el nuevo cliente en la base de datos	
Flujo Alternativo: 1. Cliente abre la ventana de registrar 2. Ingresa los datos y presiona registrar 3. Se verifican los datos en la base de datos 4. Se le muestra un mensaje de error al Cliente de algún dato erróneo o que el usuario ya existe	
Poscondiciones: Nuevo Cliente almacenado en la base de datos	

Nombre	Registrar Instructor
Autor	Jesús Andrés Chavarría Delgado
Fecha	17/04/21
Prioridad	N

Descripción: Permite Registrar un nuevo instructor en el sistema
Actores: Instructor no registrado
Precondiciones: El instructor no debe estar registrado en el sistema
Flujo Normal: 1. Instructor abre la ventana de registrar 2. Ingresa los datos y presiona el botón de registrar 3. Se verifican las credenciales 4. Se guarda el nuevo cliente en el sistema
Flujo Alternativo: 1. Instructor abre la ventana de registrar 2. Ingresa los datos y presiona el botón de registrar 3. Se verifican las credenciales 4. Se le muestra un mensaje de error al Instructor de algún dato erróneo o que las credenciales son inválidas
Poscondiciones: Nuevo Instructor registrado en el sistema

Nombre	Pagar Sesión
Autor	Jesús Andrés Chavarría Delgado
Fecha	17/04/21
Prioridad	N
Descripción: Permite a un cliente pagar el monto de una sesión	
Actores: Cliente	

Precondiciones: El Cliente debe estar registrado en el sistema y tener un pago pendiente
Flujo Normal: 1. Cliente abre la ventana de pago de sesión 2. Selecciona la sesión a cancelar y le da realizar pago 3. Se verifica el pago 4. Se actualiza la información en la base de datos 5. El cliente queda sin pagos pendientes
Flujo Alternativo: 1. Cliente abre la ventana de pago de sesión 2. Selecciona la sesión a cancelar y le da realizar pago 3. Se verifica el pago 4. Se muestra un mensaje de error de que no se pudo realizar el pago 5. Se devuelve a la ventana de pago
Poscondiciones: Se actualiza los pagos del cliente

Nombre	Cambiar Instructor
Autor	Jesús Andrés Chavarría Delgado
Fecha	17/04/21
Prioridad	N
Descripción: Cambia el Instructor de una sesión	
Actores: Administrador	
Precondiciones: Debe haber una sesión registrada con un Instructor	
Flujo Normal: 1. El administrador ingresa a la ventana de modificar sesión	

2. Selecciona a un nuevo instructor para la sesión 3. Le da al botón de realizar cambios 4. Se verifica que se pueda realizar el cambio 5. Se realizan las modificaciones en la base de datos 6. Se muestra un mensaje de que se realizó el cambio
Flujo Alternativo: 1. El administrador ingresa a la ventana de modificar sesión 2. Selecciona a un nuevo instructor para la sesión 3. Le da al botón de realizar cambios 4. Se verifica que se pueda realizar el cambio 5. No se pudo realizar el cambio 6. Se muestra un mensaje de error en la ventana
Poscondiciones: Nuevo Instructor asignado a la sesión

Nombre	Gestionar Calendario
Autor	Jesús Andrés Chavarría Delgado
Fecha	17/04/21
Prioridad	N
Descripción: Permite realizar modificaciones a los calendarios de sesiones	
Actores: Administrador	
Precondiciones: Deben haber sesiones registradas en el calendario (En caso de modificar)	
Flujo Normal: 1. El Administrador abre la ventana de gestión de Calendarios 2. Selecciona lo que desea hacer, puede ser Añadir una nueva sesión, Modificar una sesión o Eliminar una sesión 3. Se abre nueva ventana mostrando casillas según la función escogida	

4. Ingresa los datos según la opción escogida 5. Presiona el botón de Crear/Modificar/Eliminar 6. Se verifican los datos ingresados 7. Se modifica la sesion base de datos 8. Se muestra un mensaje de confirmación
Flujo Alternativo: 1. El Administrador abre la ventana de gestión de Calendarios 2. Selecciona lo que desea hacer, puede ser Añadir una nueva sesión, Modificar una sesión o Eliminar una sesión 3. Se abre nueva ventana mostrando casillas según la función escogida 4. Ingresa los datos según la opción escogida 5. Presiona el botón de Crear/Modificar/Eliminar 6. Se verifican los datos ingresados 7. Se muestra un mensaje de error en caso de que algún dato sea erróneo
Poscondiciones: Calendario modificado

Nombre	Nuevo Servicio
Autor	Jesús Andrés Chavarría Delgado
Fecha	17/04/21
Prioridad	N
Descripción: Permite crear un nuevo servicio	
Actores: Administrador	
Precondiciones: Administrador debe estar autentificado	
Flujo Normal: 1. El Administrador ingresa sus credenciales 2. Se abre la ventana de Nuevo servicio	

3. Se ingresan los datos requeridos 4. Se presiona el botón de crear 5. Se verifican los datos 6. Se guarda el nuevo servicio en la base de datos 7. Se muestra un mensaje de confirmación en la ventana
Flujo Alternativo: 1. El Administrador ingresa sus credenciales 2. Se abre la ventana de Nuevo servicio 3. Se ingresan los datos requeridos 4. Se presiona el botón de crear 5. Se verifican los datos 6. Se muestra un mensaje de que los datos ingresados son erróneos
Poscondiciones: Nuevo Servicio guardado en la base de datos

Nombre	Reservar
Autor	Jesús Andrés Chavarría Delgado
Fecha	17/04/21
Prioridad	N
Descripción: Permite a un cliente reservar en una sesión	
Actores: Cliente, Administrador	
Precondiciones: Cliente y Administrador deben estar autentificados, debe haber una sesión registrada en el sistema	
Flujo Normal: 1. Cliente abre ventana de Registrar 2. Selecciona la sesión a reservar 3. Se verifica que no tenga deudas pendientes	

4. Se verifica que haya campo en la sesión 5. Se abre ventana que muestra los métodos de pago 6. Selecciona el método de pago 7. El administrador verifica la reservación 8. Se muestra un mensaje de confirmación de la reservación
Flujo Alternativo: 1. Cliente abre ventana de Registrar 2. Selecciona la sesión a reservar 3. Se verifica que no tenga deudas pendientes 4. Se muestra un mensaje de que no puede realizar reservación hasta cancelar las deudas pendientes
Poscondiciones: Cliente registrado en la sesión

Nombre	Programar clase
Autor	Jesús Andrés Chavarría Delgado
Fecha	17/04/21
Prioridad	N
Descripción: Permite a un instructor programar una clase	
Actores: Instructor	
Precondiciones: El instructor debe estar registrado en el sistema	
Flujo Normal: 1. El instructor abre la ventana de Programar Clase 2. Ingresa los datos de la clase 3. Presiona el botón de programar 4. Se verifican los datos 5. Se guarda la clase en la base de datos	

6. Se muestra un mensaje de confirmación
Flujo Alternativo: 1. El instructor abre la ventana de Programar Clase 2. Ingresa los datos de la clase 3. Presiona el botón de programar 4. Se verifican los datos 5. Se muestra un mensaje de error en caso de que no se pueda programar la clase
Poscondiciones: Nueva clase programada

6. Vista de Procesos

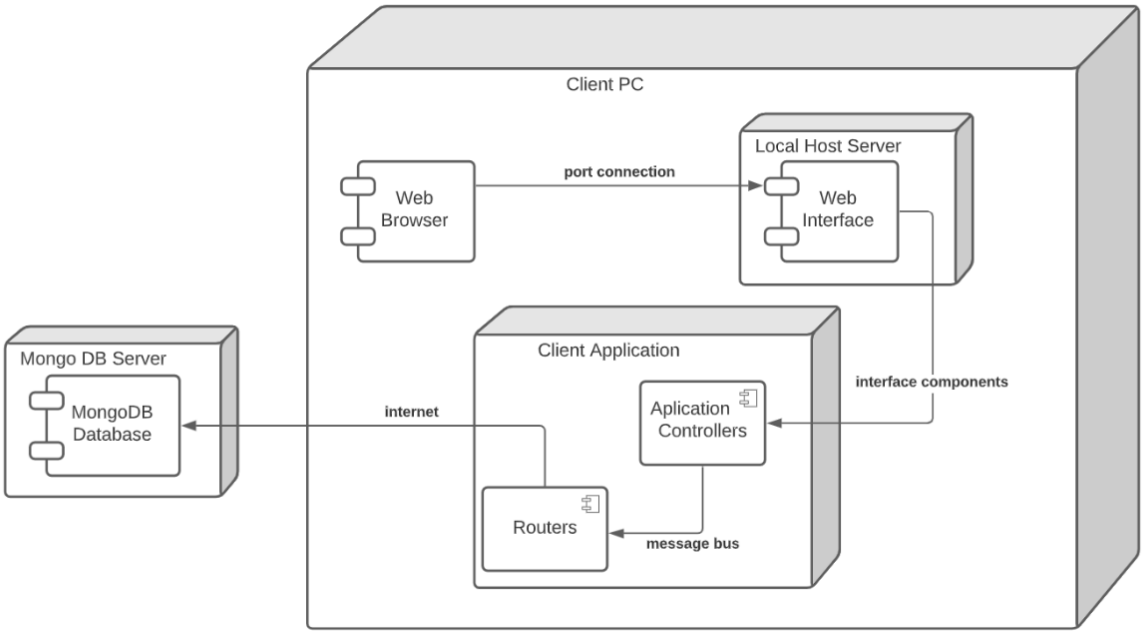
La Vista de Procesos con en el diagrama de despliegue se encuentra disponible para mejor visualización en el siguiente enlace:

https://github.com/agusbrenes/Proyecto-DIS-Gimnasio/blob/main/Diagramas/Vista%20de%20Procesos.pdf

7. Vista Física

La Vista Física está adjunta abajo en el diagrama de despliegue, y también se encuentra disponible para mejor visualización en el siguiente enlace:

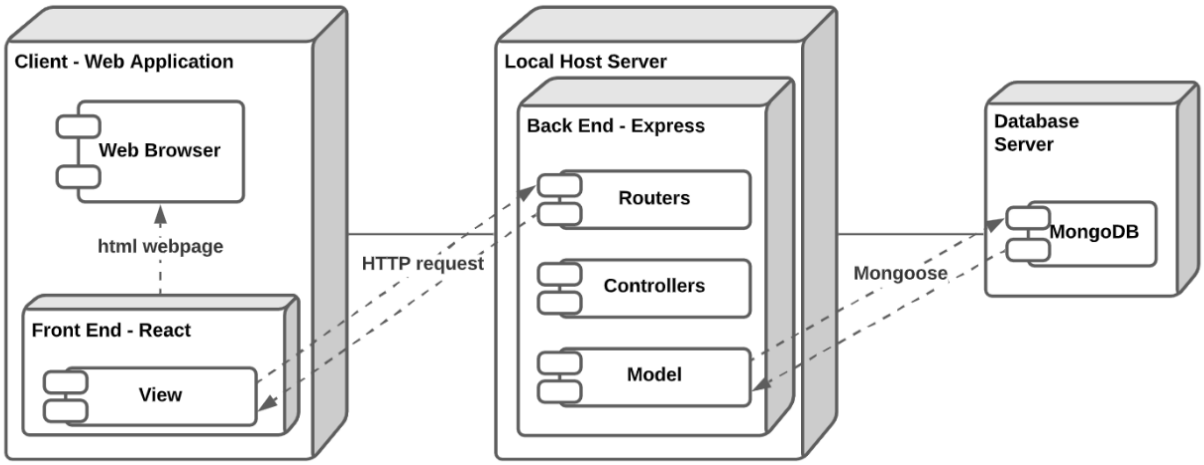
https://github.com/agusbrenes/Proyecto-DIS-Gimnasio/blob/main/Diagramas/Vista%20F%C3%ADsica.pdf



8. Vista de Desarrollo o Despliegue

La Vista de Desarrollo o de Despliegue está adjunta abajo en el diagrama de componentes, y también se encuentra disponible para mejor visualización en el siguiente enlace:

https://github.com/agusbrenes/Proyecto-DIS-Gimnasio/blob/main/Diagramas/Vista%20de%20Despliegue.pdf



8.1 Capas

8.1.1 Capa del Cliente:

- Motor de búsqueda Web
- React, Front End de la Aplicación

8.1.2 Capa del Servidor Local:

- Servidor Local
- Express, Back End de la Aplicación

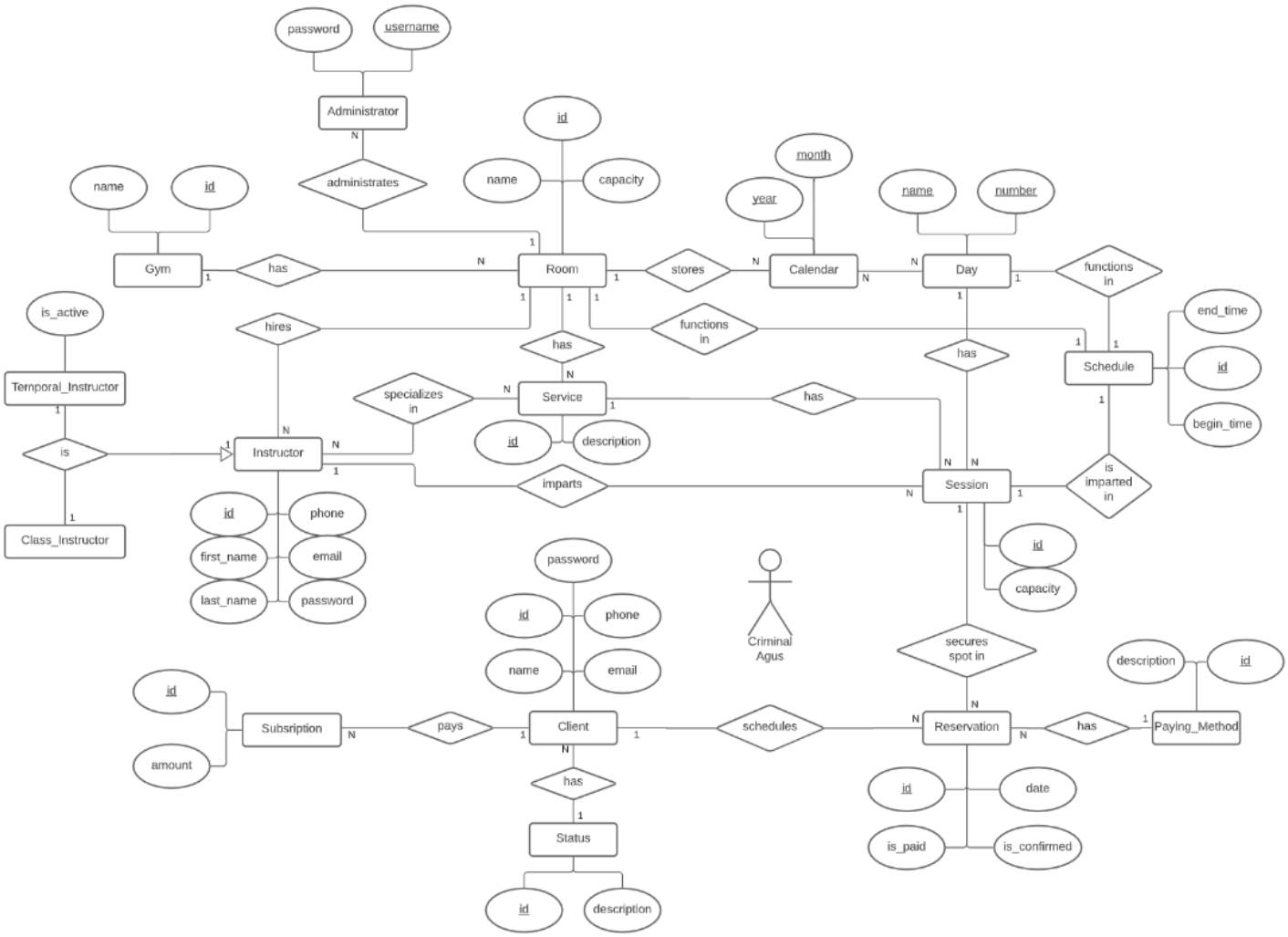
8.1.3 Capa de Base de Datos:

- MongoDB, base de datos

9. Vista de Datos

La Vista de Datos está adjunta abajo en el diagrama de entidad-relación, y también se encuentra disponible para mejor visualización en el siguiente enlace:

https://github.com/agusbrenes/Proyecto-DIS-Gimnasio/blob/main/Diagramas/Vista%20de%20Datos.pdf



10. Size and Performance

Se espera que el sistema tenga un tamaño mediano, ya que está enfocado para que sea utilizado por emprendedores medianos y pequeños, los cuales corresponden a los que han sido mayormente afectados por la pandemia del COVID-19. En lo que se refiere a mediano, se espera a que a lo mucho, existan 200 usuarios inscritos en el sistema, con aproximadamente 30 de ellos accediendo al sistema en un momento determinado.

Dicho esto, se espera que el modelado del sistema permita una implementación que sea eficiente y rápida, por lo que se espera que el servidor y el sistema no sufran de perdidas de rendimiento en cualquier momento, a no ser de que el servidor del sistema se encuentre en mantenimiento.

11. Quality

Gracias al uso de MongoDB como base de datos, se está asegurando que la base de datos se mantenga disponible las 24 horas del día durante la semana. Esto se da gracias a que el servicio de MongoDB Atlas permite crear una base de datos en la nube. Esto a su vez alivia una carga en el sistema, ya que se elimina la necesidad de que una máquina del cliente esté ejecutando el servidor de la base de datos durante todo el momento que se utilice el sistema [1].

Además, una ventaja de MongoDB es que permite ampliar el sistema a un cluster de bases de datos, por lo que permite la escalabilidad del sistema de manera fácil y efectiva [1].

Por otra parte, el realizar el sistema como una aplicación web permite que este sea inherentemente portátil, ya que cualquier máquina que realice una conexión al sistema puede acceder a este y hacer uso del mismo. Esto se logra con una mezcla de las librerías de ReactJS, NodeJS, y con el uso del lenguaje JavaScript combinado con HTML y CSS, los cuales se especializan en la creación de páginas web.

Para el guardado de contraseñas, se utilizará una función hash, lo que le agregará un aspecto de seguridad al sistema, ya que estas funciones están diseñadas para encriptar datos de manera segura.