Instituto Tecnológico de Costa Rica

Proyecto Fase 1 - Parte 2

Software Architecture Document

Version <2.0>

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
10/06/2021	2.0	Corregir errores en el documento	Eduardo

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 arquitectónico 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 se encuentra disponible para mejor visualización en el siguiente enlace:

https://github.com/agusbrenes/Proyecto-DIS-Gimnasio/blob/main/Diagramas/Diagramas%20de%20Casos%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 Controller - Database Communication

- Capa de control con el DAO pattern que se encarga de guardar y modificar objetos en la base de datos de MongoDB.

5.1.4 View

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

5.2 Algunos Casos de Uso

Nombre	RegisterClient
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21
Prioridad	N
Descripción: Este caso de uso permite a los usuarios registrarse al sistema como clientes.	
Actores: Cliente no registrado	
Precondiciones: El cliente no debe de estar registrado en la base de datos y el email y la cédula no se encuentran en uso por otro usuario.	
Flujo Normal: 1. El usuario abre la ventana de registro. 2. El usuario ingresa los datos correspondientes. 3. El usuario presiona registrarse. 4. El sistema valida los datos y genera el usuario en la base de datos. 5. El sistema despliega una ventana de registro exitoso.	
Flujo Alternativo: 4.a. El sistema detecta que los datos del email o la cédula se encuentran en uso. 5.a. El sistema despliega el mensaje de error correspondiente.	
Poscondiciones: El usuario queda registrado en el sistema como cliente.	

Nombre	RegisterInstructor
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21
Prioridad	N
Descripción:	

Este caso de uso permite a los administradores registrar instructores en el sistema
Actores: Administrador
Precondiciones: El instructor no debe de estar registrado en la base de datos y el email y la cédula no se encuentran en uso por otro usuario.
Flujo Normal: 1. El administrador abre la ventana de registro de instructor. 2. El administrador ingresa los datos correspondientes. 3. El administrador presiona para registrar al instructor. 4. El sistema valida los datos y genera el instructor en la base de datos. 5. El sistema despliega una ventana de registro exitoso.
Flujo Alternativo: 4.a. El sistema detecta que los datos del email o la cédula se encuentran en uso. 5.a. El sistema despliega el mensaje de error correspondiente.
Poscondiciones: El instructor queda registrado en el sistema con el room correspondiente.

Nombre	Login
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21
Prioridad	N
Descripción: Este caso de uso permite a los usuarios ingresar al sistema con sus credenciales correspondientes	
Actores: Usuario no autentificado.	
Precondiciones: El cliente debe estar registrado en el sistema.	
Flujo Normal:	

1. El usuario abre la ventana de login.
2. El usuario llena los credenciales correspondientes.
3. El usuario selecciona el tipo de usuario.
4. El usuario presiona login.
5. El sistema verifica los datos y le retorna un token de autorización.
6. El sistema redirecciona al usuario al menú principal.

Flujo Alternativo:

5.a. El sistema detecta que los datos del email y la contraseña no coinciden.
6.a. El sistema despliega el mensaje de error correspondiente.

Poscondiciones:
El usuario queda autenticado en el sistema.

Nombre	ManageCalendar
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21
Prioridad	N

Descripción:
Este caso permite al administrador manejar datos del calendario

Actores:
Administrador

Precondiciones:
No hay

Flujo Normal:

1. El administrador abre el calendario.
2. El administrador elige los campos a modificar.
3. El administrador le da a guardar cambios.
4. El sistema recibe los cambios y los guarda en la base de datos.
5. El sistema despliega un mensaje de aceptación.

Flujo Alternativo:

No hay

Poscondiciones: Se modifica el calendario en el sistema.

Nombre	CreateCalendar
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21
Prioridad	N
Descripción: Este caso permite al administrador crear calendarios en el sistema	
Actores: Administrador	
Precondiciones: Debe de existir un room en el sistema.	
Flujo Normal: 1. El administrador abre el menú del room. 2. El administrador elige crear un calendario. 3. El administrador llena los datos del calendario. 4. El administrador presiona crear calendario. 5. El sistema crea el calendario y lo guarda en el sistema. 6. El sistema muestra un mensaje de operación exitosa.	
Flujo Alternativo: No hay	
Poscondiciones: Se crea un calendario en el sistema.	

Nombre	ViewCalendars
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21

Prioridad	N
Descripción: Este caso permite a los usuarios ver los calendarios del room.	
Actores: Administrador y Cliente	
Precondiciones: Debe de existir un room en el sistema.	
Flujo Normal: 1. El usuario abre el menú del room. 2. El usuario elige ver los calendarios. 3. El sistema despliega la lista de calendarios.	
Flujo Alternativo: No hay	
Poscondiciones: Se muestran los calendarios en el sistema.	

Nombre	CheckServices
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21
Prioridad	N
Descripción: Este caso permite a los clientes la lista de servicios.	
Actores: Cliente	
Precondiciones: No hay.	
Flujo Normal: 1. El usuario abre el menú principal y selecciona ver lista de servicios. 2. El sistema despliega la lista de servicios del sistema.	

Flujo Alternativo: No hay
Poscondiciones: Se muestran los servicios del sistema.

Nombre	CheckReservations
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21
Prioridad	N
Descripción: Este caso permite a los clientes la lista de reservaciones	
Actores: Cliente	
Precondiciones: No hay.	
Flujo Normal: 1. El usuario abre el menú principal y selecciona ver lista de reservaciones. 2. El sistema despliega la lista de reservaciones creadas por el cliente.	
Flujo Alternativo: No hay	
Poscondiciones: Se muestran las reservaciones del sistema.	

Nombre	InstructorList
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21
Prioridad	N

Descripción: Este caso permite a los administradores ver la lista de instructores.
Actores: Administrador
Precondiciones: No hay.
Flujo Normal: 1. El administrador abre el menú principal y elige ver la lista de instructores. 2. El sistema despliega la lista de instructores que hay en el sistema.
Flujo Alternativo: No hay
Poscondiciones: Se muestran los instructores del sistema.

Nombre	ClientList
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21
Prioridad	N
Descripción: Este caso permite a los administradores ver la lista de los clientes.	
Actores: Administrador	
Precondiciones: No hay.	
Flujo Normal: 1. El administrador abre el menú principal y elige ver la lista de clientes. 2. El sistema despliega la lista de clientes registrados en el sistema	
Flujo Alternativo: No hay	

Poscondiciones: Se muestran a los clientes del sistema.

Nombre	ServiceList
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21
Prioridad	N
Descripción: Este caso permite a los administradores ver la lista de los servicios	
Actores: Administrador	
Precondiciones: No hay.	
Flujo Normal: 1. El administrador abre el menú principal y elige ver la lista de servicios. 2. El sistema despliega la lista de servicios del sistema.	
Flujo Alternativo: No hay	
Poscondiciones: Se muestran los servicios del sistema.	

Nombre	SessionList
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21
Prioridad	N
Descripción: Este caso permite a los administradores ver la lista de las sesiones creadas por los instructores	

Actores: Administrador
Precondiciones: No hay.
Flujo Normal: 1. El administrador abre el menú principal y elige ver la lista de sesiones. 2. El sistema despliega la lista de sesiones.
Flujo Alternativo: No hay
Poscondiciones: Se muestran las sesiones registradas en el sistema.

Nombre	PaySubscription
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21
Prioridad	N
Descripción: Este caso permite a los clientes pagar las suscripciones que han hecho,	
Actores: Client	
Precondiciones: Hay suscripciones sin pagar.	
Flujo Normal: 1. El cliente elige la opción de ver las suscripciones sin pagar. 2. El sistema muestra las suscripciones sin pagar en la pantalla. 3. El cliente elige una suscripción a pagar. 4. El cliente selecciona el método de pago. 5. El cliente selecciona pagar. 6. El sistema guarda las modificaciones y despliega el mensaje en pantalla.	
Flujo Alternativo:	

No hay
Poscondiciones: Se guarda como suscripción pagada en el sistema.

Nombre	PayReservation
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21
Prioridad	N
Descripción: Este caso permite a los clientes pagar las reservaciones que han hecho,	
Actores: Client	
Precondiciones: Hay reservaciones sin pagar.	
Flujo Normal: 1. El cliente elige la opción de ver las reservaciones sin pagar. 2. El sistema muestra las reservaciones sin pagar en la pantalla. 3. El cliente elige una reservación a pagar. 4. El cliente selecciona el método de pago. 5. El cliente selecciona pagar. 6. El sistema guarda las modificaciones y despliega el mensaje en pantalla.	
Flujo Alternativo: No hay	
Poscondiciones: Se guarda como reservación pagada en el sistema.	

Nombre	RegisterPayMethod
Autor	Eduardo Josué Saborío Pérez

Fecha	10/06/21
Prioridad	N
Descripción: Este caso permite a los administradores registrar nuevas formas de pago para que los clientes las utilicen,	
Actores: Administrador	
Precondiciones: No hay	
Flujo Normal: 1. El administrador llena los campos del método de pago nuevo. 2. El sistema guarda los datos. 3. El sistema despliega en pantalla un mensaje de aceptación.	
Flujo Alternativo: No hay	
Poscondiciones: Se guarda como nuevo método de pago en el sistema.	

Nombre	NewService
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21
Prioridad	N
Descripción: Este caso permite a los administradores crear nuevos servicios para un room.	
Actores: Administrador	
Precondiciones: Existe un room en el sistema	

Flujo Normal: 1. El administrador elige crear un servicio nuevo. 2. El administrador llena los datos y presiona registrar servicio. 3. El sistema valida los datos y los guarda en la base de datos. 4. El sistema muestra en pantalla el mensaje de aceptación
Flujo Alternativo: No hay
Poscondiciones: Se guarda como un nuevo servicio en el sistema.

Nombre	RegisterSpeciality
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21
Prioridad	N
Descripción: Este caso permite a los instructores registrar las especialidades que tienen.	
Actores: Instructor	
Precondiciones: Existe un room en el sistema y también existen servicios.	
Flujo Normal: 1. El instructor elige la opción de agregar especialidad. 2. El instructor elige la especialidad a agregar. 3. El instructor manda la solicitud. 4. El sistema modifica el instructor para agregarle la especialidad. 5. El sistema muestra en pantalla un mensaje de aceptación	
Flujo Alternativo: No hay	
Poscondiciones: Se modifica el instructor en el sistema para agregar su nueva especialidad.	

Nombre	ModifySession
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21
Prioridad	N
Descripción: Este caso permite a los administradores modificar las sesiones existentes en el sistemas.	
Actores: Administrador	
Precondiciones: Existe una sesión en el sistema.	
Flujo Normal: 1. El administrador elige ver la lista de sesiones. 2. El sistema despliega la lista de sesiones. 3. El administrador selecciona una sesión y las modificaciones a realizar. 4. El sistema guarda las modificaciones en el sistema y despliega un mensaje de aceptación.	
Flujo Alternativo: No hay	
Poscondiciones: Se modifica la sesión en el sistema.	

Nombre	DeleteSession
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21
Prioridad	N
Descripción: Este caso permite a los administradores borrar las sesiones existentes en el sistemas.	

Actores: Administrador
Precondiciones: Existe una sesión en el sistema.
Flujo Normal: 1. El administrador elige ver la lista de sesiones. 2. El sistema despliega la lista de sesiones. 3. El administrador selecciona una sesión y solicita borrarla. 4. El sistema borra la sesión del sistema y despliega un mensaje de aceptación.
Flujo Alternativo: No hay
Poscondiciones: Se elimina la sesión en el sistema.

Nombre	CreateSession
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21
Prioridad	N
Descripción: Este caso permite a los instructores crear sesiones en el sistema	
Actores: Instructor	
Precondiciones: Existe el servicio en el sistema.	
Flujo Normal: 1. El instructor elige crear una nueva sesión. 2. El instructor elige el servicio del que trata la sesión. 3. El instructor elige la fecha (tiene que comprobar que hay espacio). 4. El sistema valida los datos y guarda la nueva sesión. 5. El sistema despliega un mensaje de aceptación.	

Flujo Alternativo: No hay
Poscondiciones: Se agenda una nueva sesión en el sistema.

Nombre	ReserveSession
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21
Prioridad	N
Descripción: Este caso permite a los clientes crear una reservación de una sesión.	
Actores: Cliente	
Precondiciones: Existe la sesión en el sistema	
Flujo Normal: 1. El cliente elige crear una nueva reservación. 2. El cliente elige la sesión a reservar. 3. El sistema valida los datos y crea la reservación en el sistema. 4. El sistema despliega un mensaje de aceptación.	
Flujo Alternativo: No hay	
Poscondiciones: Se agenda una nueva reservación no aprobada en el sistema.	

Nombre	ConfirmReservation
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21

Prioridad	N
Descripción: Este caso permite a los administradores aprobar reservaciones en el sistema.	
Actores: Administrator	
Precondiciones: Existe la reservación sin aprobar en el sistema.	
Flujo Normal: 1. El administrador presiona ver la lista de reservaciones sin aprobar. 2. El sistema despliega la lista de reservaciones sin aprobar. 3. El administrador elige la reservación a aprobar y manda la solicitud. 4. El sistema modifica la reserva para que quede aprobada y despliega un mensaje de aceptación en el sistema.	
Flujo Alternativo: No hay	
Poscondiciones: Se aprueba la reservación en el sistema.	

Nombre	CreateRoom
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21
Prioridad	N
Descripción: Este caso permite a los administradores crear nuevos rooms en el sistema.	
Actores: Administrator	
Precondiciones: No hay.	
Flujo Normal:	

1. El administrador presiona la opción de crear un room. 2. El administrador llena los datos correspondientes del room. 3. El sistema valida los datos y crea el registro en el sistema. 4. El sistema muestra un mensaje de aceptación.
Flujo Alternativo: No hay
Poscondiciones: Se crea un nuevo room en el sistema.

Nombre	ModifyRoom
Autor	Eduardo Josué Saborío Pérez
Fecha	10/06/21
Prioridad	N
Descripción: Este caso permite a los administradores modificar rooms en el sistema.	
Actores: Administrator	
Precondiciones: Existen rooms en el sistema.	
Flujo Normal: 1. El administrador presiona la opción de ver la lista de rooms de los cuales es administrador. 2. El administrador presiona la opción de modificar el room. 3. El administrador llena los campos nuevos y envía la solicitud. 4. El sistema valida los datos y modifica el registro en el sistema. 5. El sistema muestra un mensaje de aceptación.	
Flujo Alternativo: No hay	
Poscondiciones: Se modifica un room en el sistema.	

6. Vista de Procesos

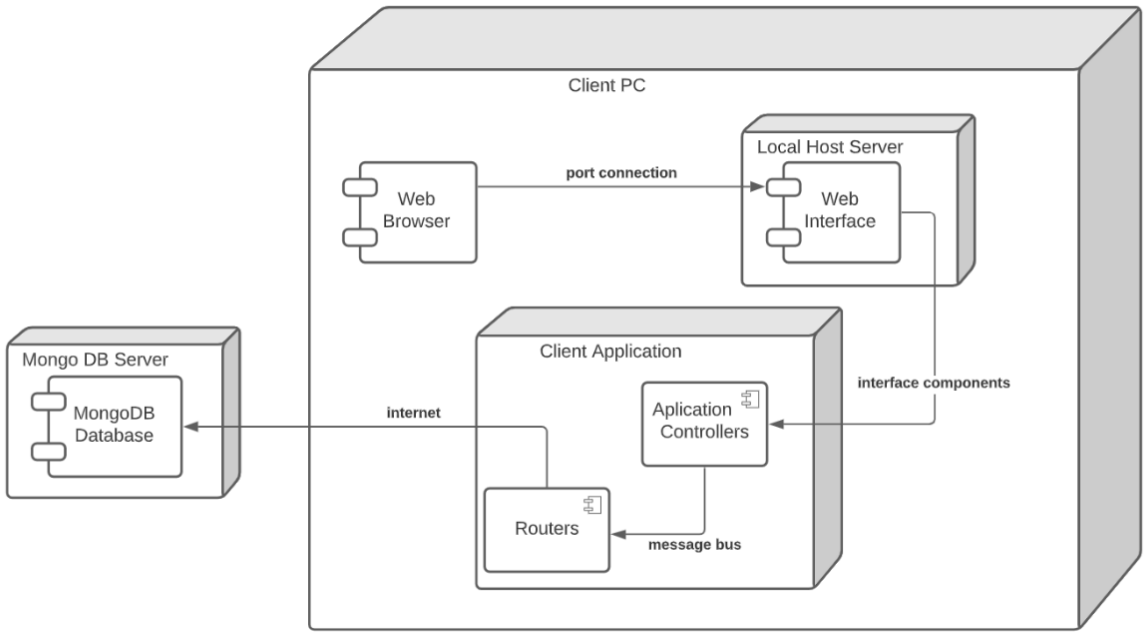
La Vista de Procesos con en el diagrama de despliegue se encuentra disponible para mejor visualización en el siguiente enlace:

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

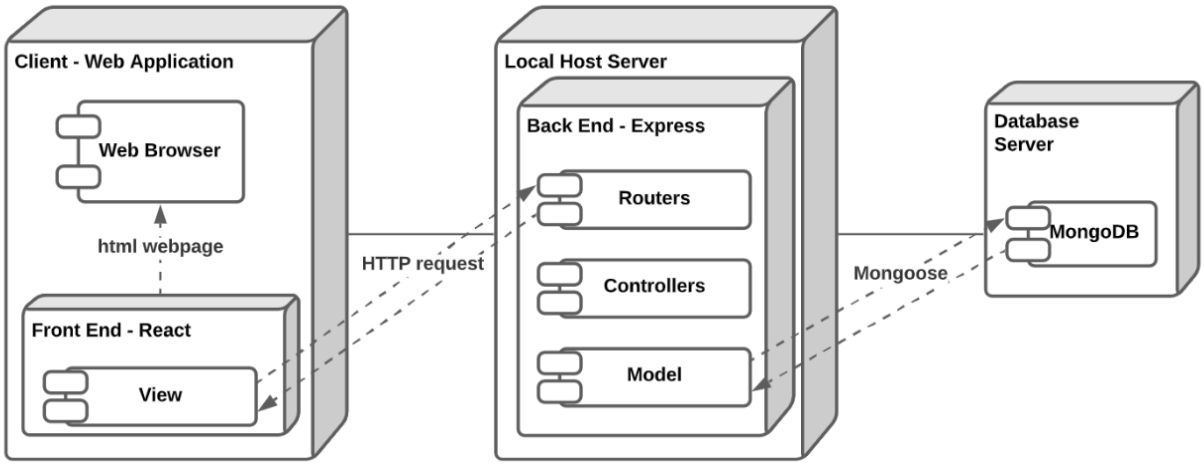


8. Vista de Desarrollo o Despliegue

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

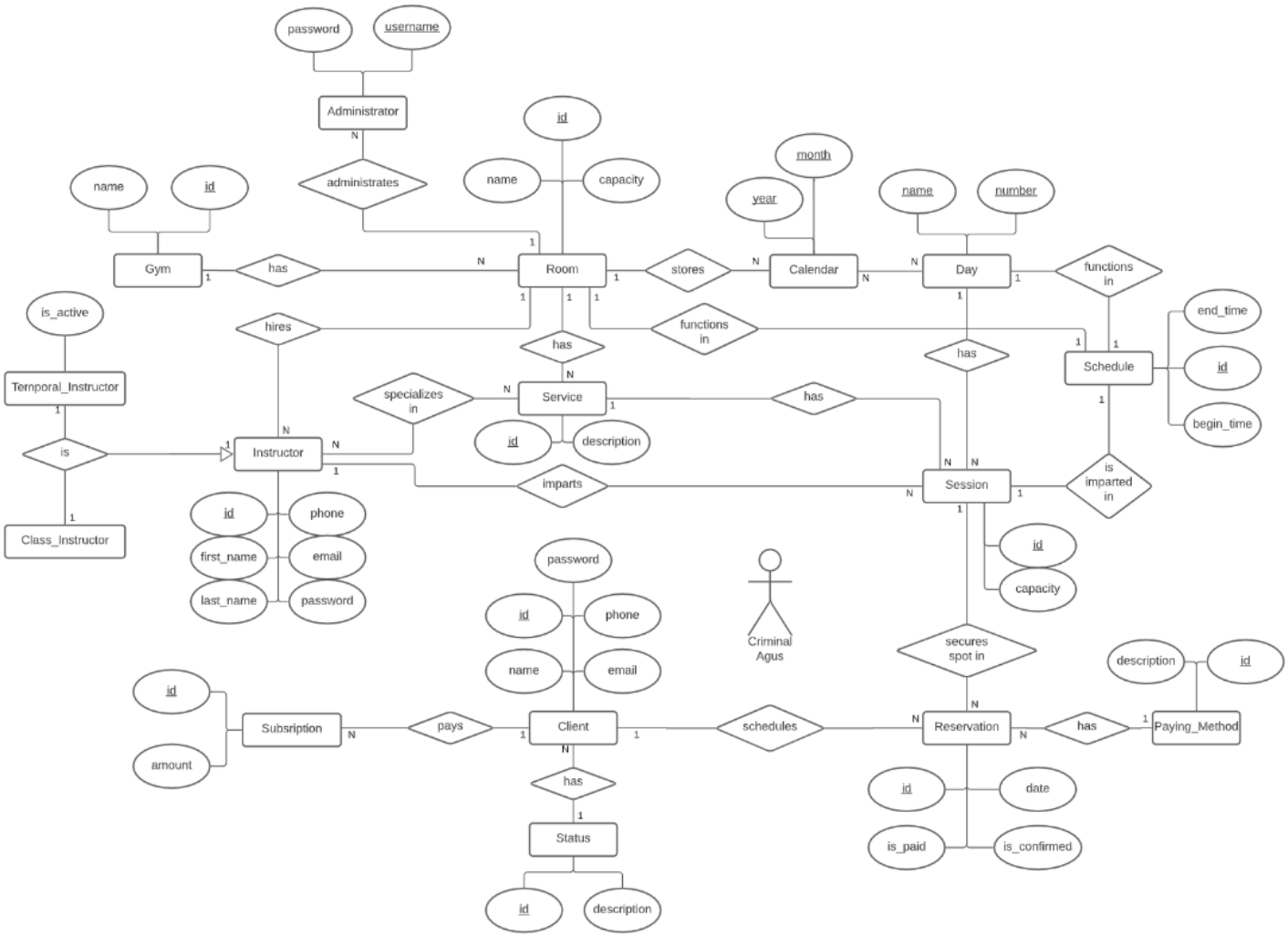
8.1.3 Capa de Base de Datos:

- 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.