

```

    } { // loads controller
    $controller = $this->request[0];
    if (class_exists($controller)) {
        $controller = new $controller(); // creates an instance of this controller
        $this->request[1] = !$this->request[1]?"index":$this->request[1]; // index 1
        $method = $this->request[1];
        $method = str_replace("-", "_", $method); // replaces hifen on url by underling
        $method = ( !method_exists($controller, $method) && (!Config::$indexMethod)
        if (method_exists($controller, $method)) {
            $firstParam = ($method == "index" && ($this->request[1] != "index") ? 1
            for ($i = $firstParam; ($i < count($this->request)) && (($i - $firstParam

```

TRABAJO INTEGRADOR

Tecnicatura en Sistemas IT



Cámara de Industria y Comercio
Argentino-Alemana
Deutsch-Argentinische
Industrie- und Handelskammer

Lo hacemos posible.

PRIMER AÑO

Indice

❑ Sistema de Automatización de Piscinas (SAP)	3
❑ Especificación de los Requerimientos	4
❑ Anforderungs-spezifikation	7
❑ Requirements specification	10
❑ Diagrama General de Casos de Usos	13
❑ Metodología a utilizar	14
❑ Cascada	14
❑ Kanban	14
❑ Roles necesarios	14
❑ Diagrama de Gantt	15
❑ Tareas	15
❑ Diagrama	16
❑ Diagrama Entidad-relación	17
❑ Modelo de Datos Relacional	18
❑ Diagrama de Clases	19
❑ Maqueta de Interfaz de Usuario	20
❑ Indice	20
❑ Justificación	20
❑ Maqueta	21
❑ Listado de Componentes	24
❑ Protocolos de Comunicación	24
❑ Diagrama de Arquitectura de Solución de Alto Nivel	24
❑ Programa en Java	25

Sistema de Automatización de Piscinas (SAP)

Una red de gimnasios nos contrató para diseñar e implementar una solución IoT que regule automáticamente la temperatura y la acidez del agua, optimizando la eficiencia de recursos energéticos.

El objetivo del proyecto consiste en implementar un sistema que mida la temperatura del agua y en función de un umbral configurable, active el suministro de agua climatizada. A su vez, el sistema deberá medir el nivel de acidez (PH) del agua y también, según valores configurables, suministrar una cierta cantidad de cloro. Como todas las sedes cuentan con acceso a la azotea, el proyecto también incluye la instalación y puesta en servicio de termotanques solares, de forma tal de reducir el consumo de gas natural. De esta manera, si la piscina necesita incorporar agua caliente y la temperatura del agua en el termotanque solar es adecuada, se evitará encender la caldera a gas.

El Usuario gestionará el sistema desde una interfaz web que debe adaptarse al dispositivo. Para cada gimnasio hay 3(tres) tipos de perfiles: Administrador, Configurador y Visualizador. El Administrador podrá gestionar (alta, baja, modificación) de usuarios, sensores y reglas. Además visualizará el estado de los sensores en tiempo real o un gráfico con valores históricos que podrá filtrarlos por fecha y hora. El Configurador podrá configurar reglas y visualizar el estado de los sensores. El Visualizador simplemente verá el estado de los sensores. La reglas deben permitir asociar un sensor y un valor, y en caso de cumplirse generar una acción (por ejemplo encender la caldera). Cada acción puede tener una alerta asociada que tendrá un usuario configurado de recibirla. Un mismo usuario tendrá la capacidad de gestionar varios gimnasios. Se debe almacenar el valor de cada sensor cada un minuto conociendo, el sensor, la pileta a la que corresponde y el gimnasio dónde está ubicado. El sistema deberá tener un usuario superadministrador, que permite además gestionar los gimnasios. Este usuario tendrá una vista adicional dónde verá un mapa con la ubicación de cada gimnasio y los parámetros principales de los mismos.

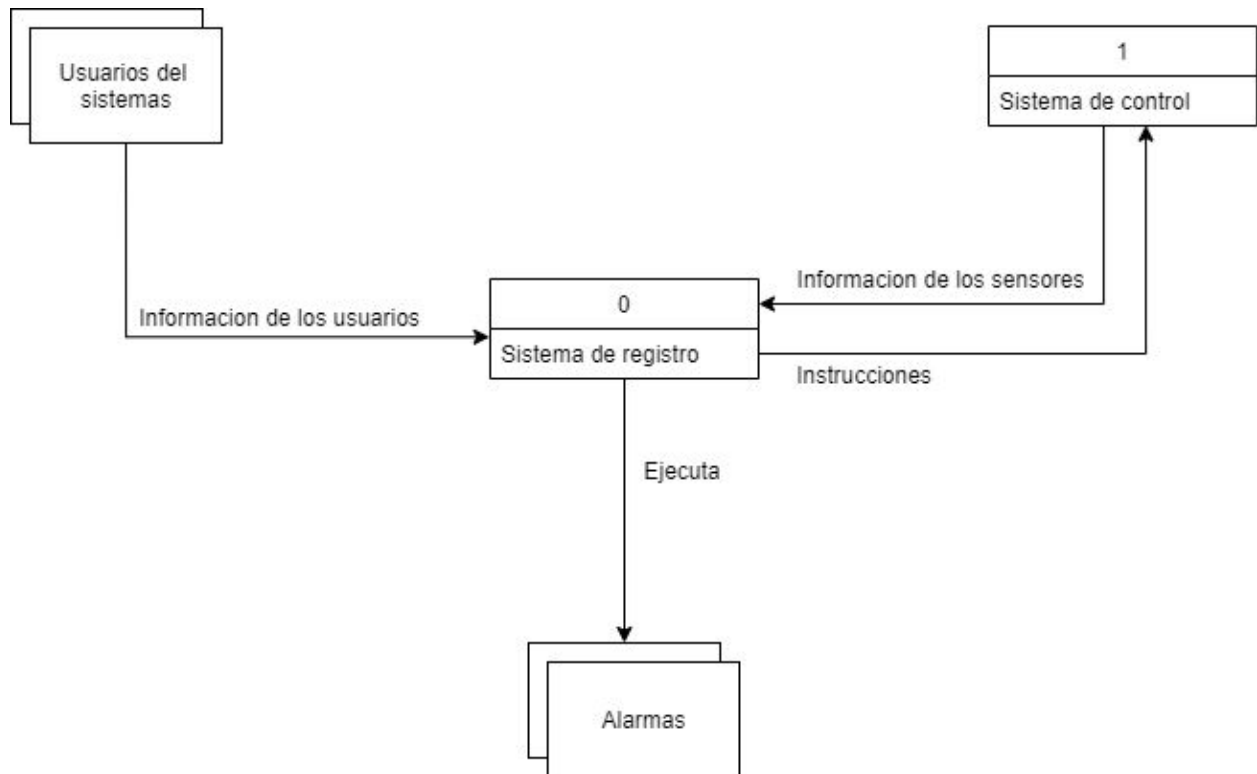
Especificación de los requerimientos

Se detallan a continuación los requerimientos funcionales del sistema.

- **Administración gimnasios**
 - **CRUD Usuarios**
 - Create
 - Nombre (VARCHAR)
 - Contraseña (VARCHAR)
 - Mail (VARCHAR)
 - Telefono (VARCHAR)
 - Read
 - Update
 - Delete
 - **CRUD Gimnasios**
 - Create
 - Nombre (VARCHAR)
 - Localización (VARCHAR)
 - Gerenteld (INT)
 - EmpleadosId (INT)
 - Read
 - Update
 - Delete
 - **CRUD Sensores**
 - Create
 - Nombre (VARCHAR)
 - Lugar (VARCHAR)
 - LimitesId (INT)
 - SistemaDeMedición (VARCHAR)
 - Read
 - Update
 - Delete
 - **CRUD Reglas**
 - Create
 - Nombre (VARCHAR)
 - Descripción (VARCHAR)
 - LimitesId (INT)
 - AccionesId (INT)
 - UsuariosId (INT)
 - Read
 - Update

Lo hacemos posible.

- Delete
 - Recuperación de contraseña
 - Verificación de usuario
 - Consulta de Log y actividad
 - Usuario administrador
- Sistema de control de piletas
 - Alarma para los usuarios suscritos
 - Histórico de los valores de los sensores
 - Ejecución de las acciones asociadas
 - Reporte por gimnasio
- Diagrama de contexto



Lo hacemos posible.

- **Especificación de requisitos**

Requisito	Administración de gimnasios y usuarios
Tipo	Funcional
Fundamento	Los usuarios deben poder acceder al sistema y ver el estado de su gimnasios correspondientes
Volatilidad	Baja probabilidad de cambio
Prioridad	Alta
Criticidad	Obligatorio
Factibilidad	Posible de implementar
Riesgo	Bajo riesgo

Requisito	Control del estado de las piletas
Tipo	Funcional
Fundamento	Los parámetros medidos por los sensores deben mantenerse siempre dentro de ciertos límites establecidos
Volatilidad	Baja probabilidad de cambio
Prioridad	Alta
Criticidad	Obligatorio
Factibilidad	Posible de implementar
Riesgo	Bajo riesgo

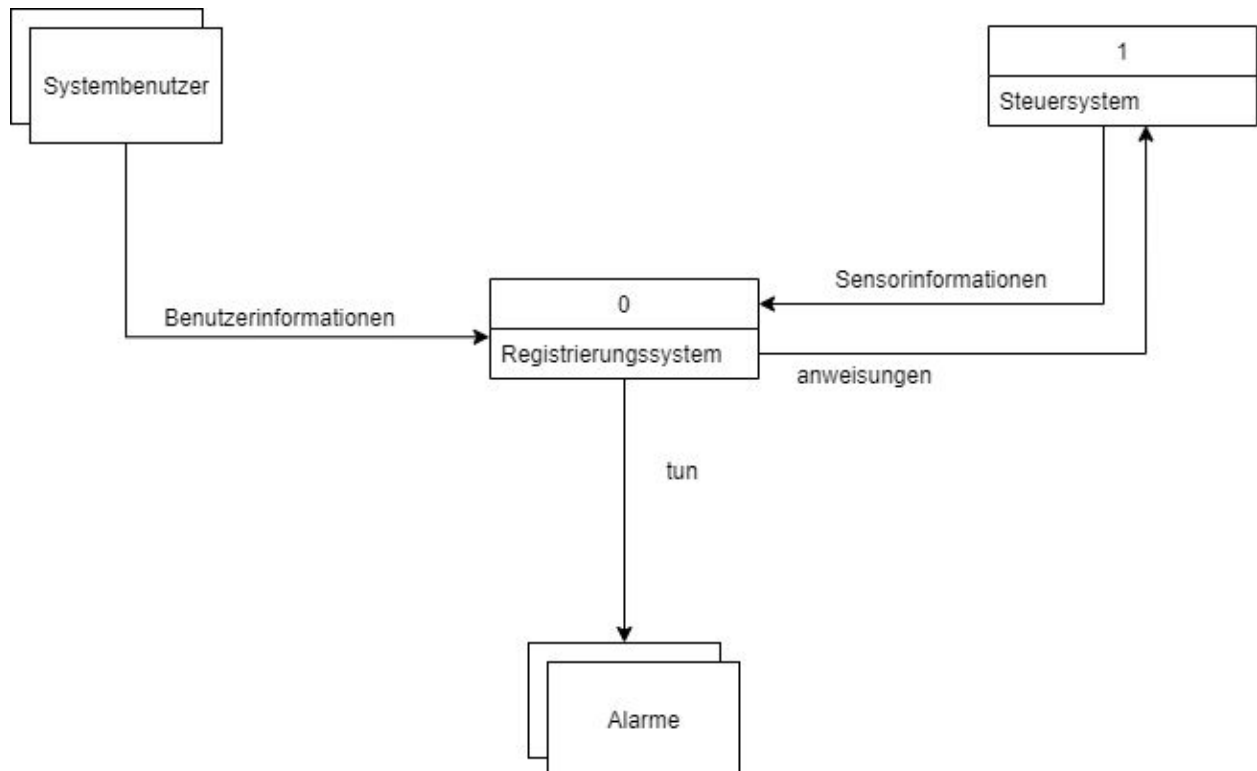
Anforderungsspezifikation

Die funktionalen Anforderungen des Systems sind nachstehend aufgeführt

- **Fitnessstudio-Verwaltung**
 - **CRUD benutzer**
 - Create
 - Name (VARCHAR)
 - Passwort (VARCHAR)
 - EMail (VARCHAR)
 - Telefon (VARCHAR)
 - Read
 - Update
 - Delete
 - **CRUD Fitnessstudio**
 - Create
 - Name (VARCHAR)
 - Lage (VARCHAR)
 - ManagerId (INT)
 - MitarbeiterId (INT)
 - Read
 - Update
 - Delete
 - **CRUD Sensoren**
 - Create
 - Name (VARCHAR)
 - platzieren (VARCHAR)
 - GrenzenId (INT)
 - Messsystem (VARCHAR)
 - Read
 - Update
 - Delete
 - **CRUD Regeln**
 - Create
 - Name (VARCHAR)
 - beschreibung (VARCHAR)
 - GrenzenId (VARCHAR)
 - Handlungen (VARCHAR)
 - Benutzername (VARCHAR)
 - Read
 - Update

Lo hacemos posible.

- Delete
 - Passwort-Wiederherstellung
 - Benutzer Überprüfung
 - Ahrung von Log und Aktivität
 - Administrator Benutzer
- Pool-Kontrollsystem
 - Alarm für abonnierte benutzer
 - Historie der sensor werte
 - Ausführung der zugehörigen Aktionen
 - Fitnessbericht
- Kontextdiagramm



- **Anforderungsspezifikation**

Anforderung	Fitnessraum und Benutzerverwaltung
Typ	funktionell
Gegrundet	Benutzer müssen in der Lage sein, auf das System zuzugreifen und den Status ihrer entsprechenden Fitnessstudios anzuzeigen
Volatilitaet	geringe wahrscheinlichkeit einer veränderung
Prioritat haben	hoch
Kritikalitat	verpflichtend
Machbarkeit	möglich zu implementieren
Risiko	geringes risiko

Anforderung	Kontrolle des Zustands der Pools
Typ	funktionell
Gegrundet	Die von den Sensoren gemessenen Parameter müssen immer innerhalb bestimmter festgelegter Grenzen gehalten werden
Volatilitaet	geringe wahrscheinlichkeit einer veränderung
Prioritat haben	hoch
Kritikalitat	verpflichtend
Machbarkeit	möglich zu implementieren
Risiko	geringes risiko

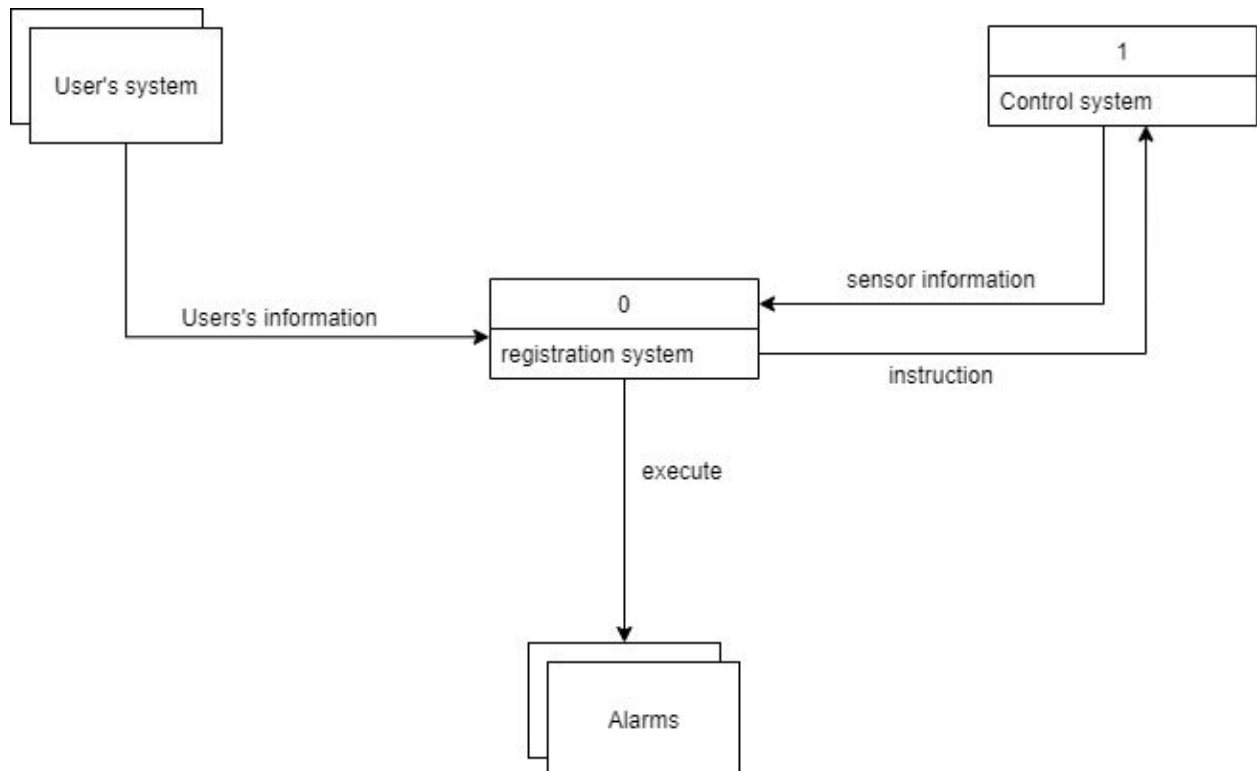
requirements specification

the system's requirements functions are detailed below

- **Gym's administration**
 - **CRUD Users**
 - Create
 - Name (VARCHAR)
 - Password (VARCHAR)
 - EMail (VARCHAR)
 - Phone (VARCHAR)
 - Read
 - Update
 - Delete
 - **CRUD gyms**
 - Create
 - Name (VARCHAR)
 - Localization (VARCHAR)
 - ManagerId (INT)
 - CustomersId (INT)
 - Read
 - Update
 - Delete
 - **CRUD Sensors**
 - Create
 - Name (VARCHAR)
 - place (VARCHAR)
 - LimitsId (INT)
 - measurementSystem (VARCHAR)
 - Read
 - Update
 - Delete
 - **CRUD Rules**
 - Create
 - Name (VARCHAR)
 - Description (VARCHAR)
 - LimitsId (VARCHAR)
 - ActionsId(VARCHAR)
 - UsersId(VARCHAR)
 - Read
 - Update

Lo hacemos posible.

- Delete
 - Password recovery
 - User verification
 - Log query and activity
 - User administrator
- Pool's control system
 - Alarm for subscribed users
 - historical sensors value
 - Execution of associated action
 - Gym reports
- Context diagram



Lo hacemos posible.

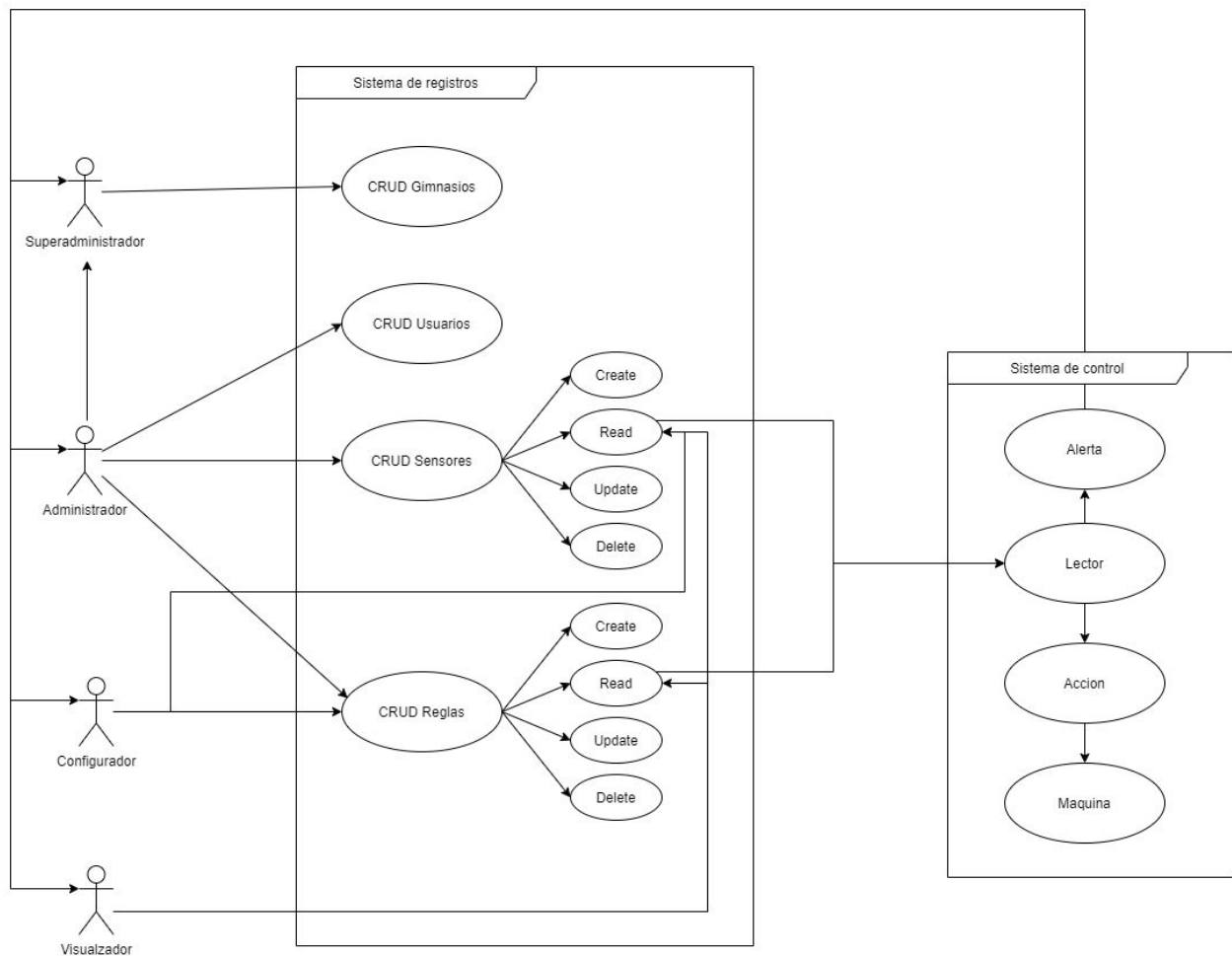
- **requirements specification**

required	Gyms and users administrations
Type	Functional
Fundamentals	The users must can access to the system and see his gyms states
Volatility	Low probability of change
Priority	High
criticality	Required
Factibility	Possible implementation
Risk	Low risk

required	Pool's control
Type	Functional
Fundamentals	The parameter measure by sensors must maintain ever inside of set limits
Volatility	Low probability of change
Priority	High
criticality	Required
Factibility	Possible implementation
Risk	Low risk

Lo hacemos posible.

Diagrama general de casos de uso



Metodología a utilizar

Cascada

Debido a la magnitud del proyecto y su naturaleza sería necesario evitar la repetición de tareas y procesos para agilizar los tiempos. Es por ello que una metodología en cascada es lo más adecuada. De esta forma se evita perder tiempo en rediseñar, reestructurar y construir nuevamente de cero el software. Una planeación meticulosa del proyecto evitará problemas a futuro y facilitará el trabajo.

Kanban

En conjunto con la anterior metodología se podría utilizar kanban para hacer un seguimiento de las tareas a realizar y las que ya fueron realizadas para evitar demoras en la evaluación del avance. De esta forma se puede obtener información fiable y rápida sobre la evolución del proyecto y la etapa en la que se encuentra. Además de todo esto el método kanban evita cuellos de botella y permite una entrega constante para hacer verificaciones con el cliente.

Roles necesarios

- Diseñador web
 - Diseñar interfaz de usuarios
- Arquitecto de base de datos
 - Diseñar base de datos
 - Implementar la base de datos
 - Mantenimiento de la base de datos
- Programador de POO
 - Diseñar el sistema backend
 - Implementar la solución
 - Mantenimiento del software
- Programador de API
 - Diseñar APIs de conexión entre la DB y la interfaz
 - Diseñar APIs de conexión entre el software backend y la interfaz
 - Implementar solución

Lo hacemos posible.

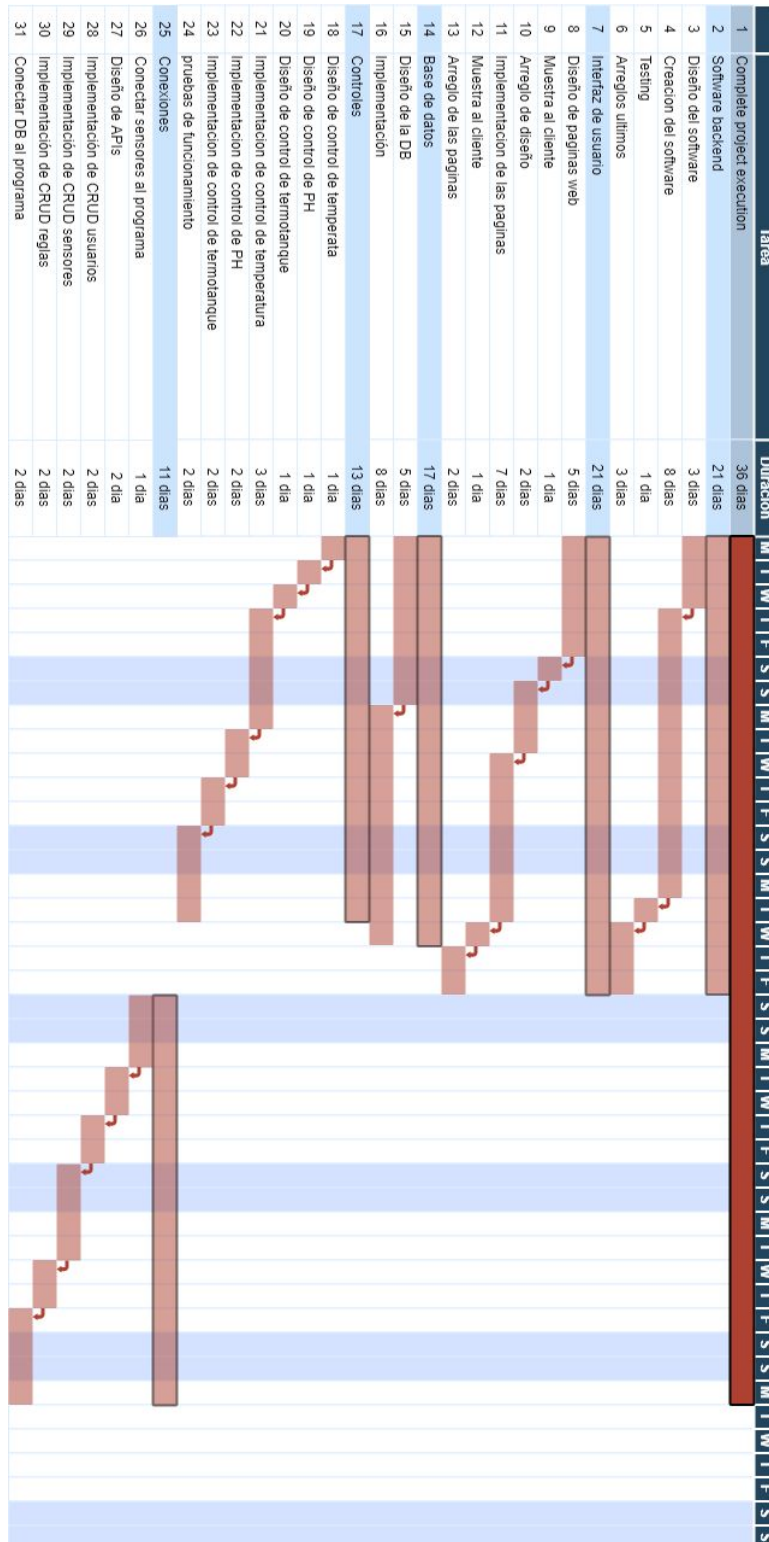
Diagrama de Gantt

Tareas

	Tarea	Duración
1	Complete project execution	36 dias
2	Software backend	21 dias
3	Diseño del software	3 dias
4	Creacion del software	8 dias
5	Testing	1 dia
6	Arreglos ultimos	3 dias
7	Interfaz de usuario	21 dias
8	Diseño de paginas web	5 dias
9	Muestra al cliente	1 dia
10	Arreglo de diseño	2 dias
11	Implementacion de las paginas	7 dias
12	Muestra al cliente	1 dia
13	Arreglo de las paginas	2 dias
14	Base de datos	17 dias
15	Diseño de la DB	5 dias
16	Implementación	8 dias
17	Controles	13 dias
18	Diseño de control de temperata	1 dia
19	Diseño de control de PH	1 dia
20	Diseño de control de termotanque	1 dia
21	Implementacion de control de temperatura	3 dias
22	Implementacion de control de PH	2 dias
23	Implementacion de control de termotanque	2 dias
24	pruebas de funcionamiento	2 dias
25	Conexiones	11 dias
26	Conectar sensores al programa	1 dia
27	Diseño de APIs	2 dia
28	Implementación de CRUD usuarios	2 dias
29	Implementación de CRUD sensores	2 dias
30	Implementación de CRUD reglas	2 dias
31	Conectar DB al programa	2 dias

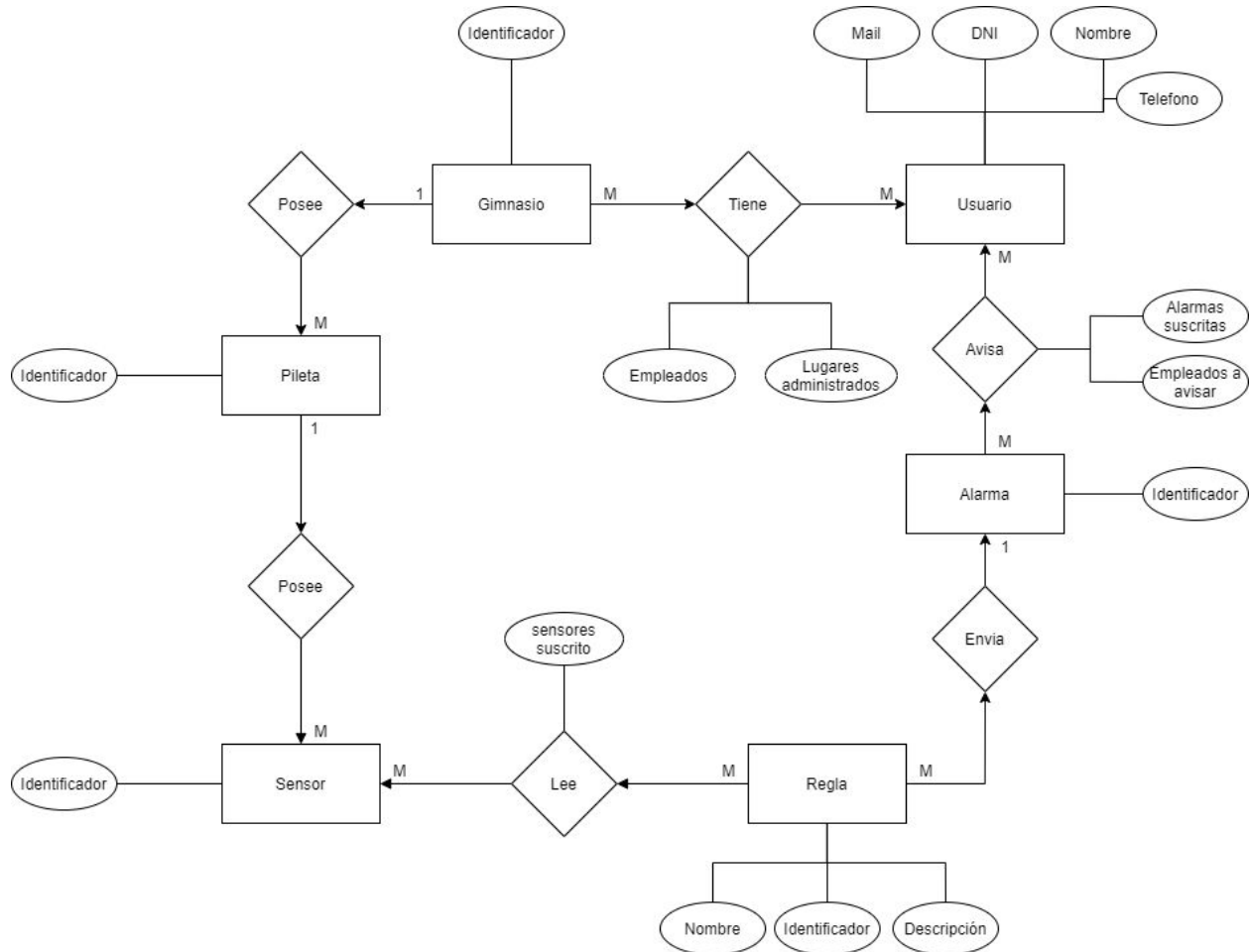
Lo hacemos posible.

Diagrama



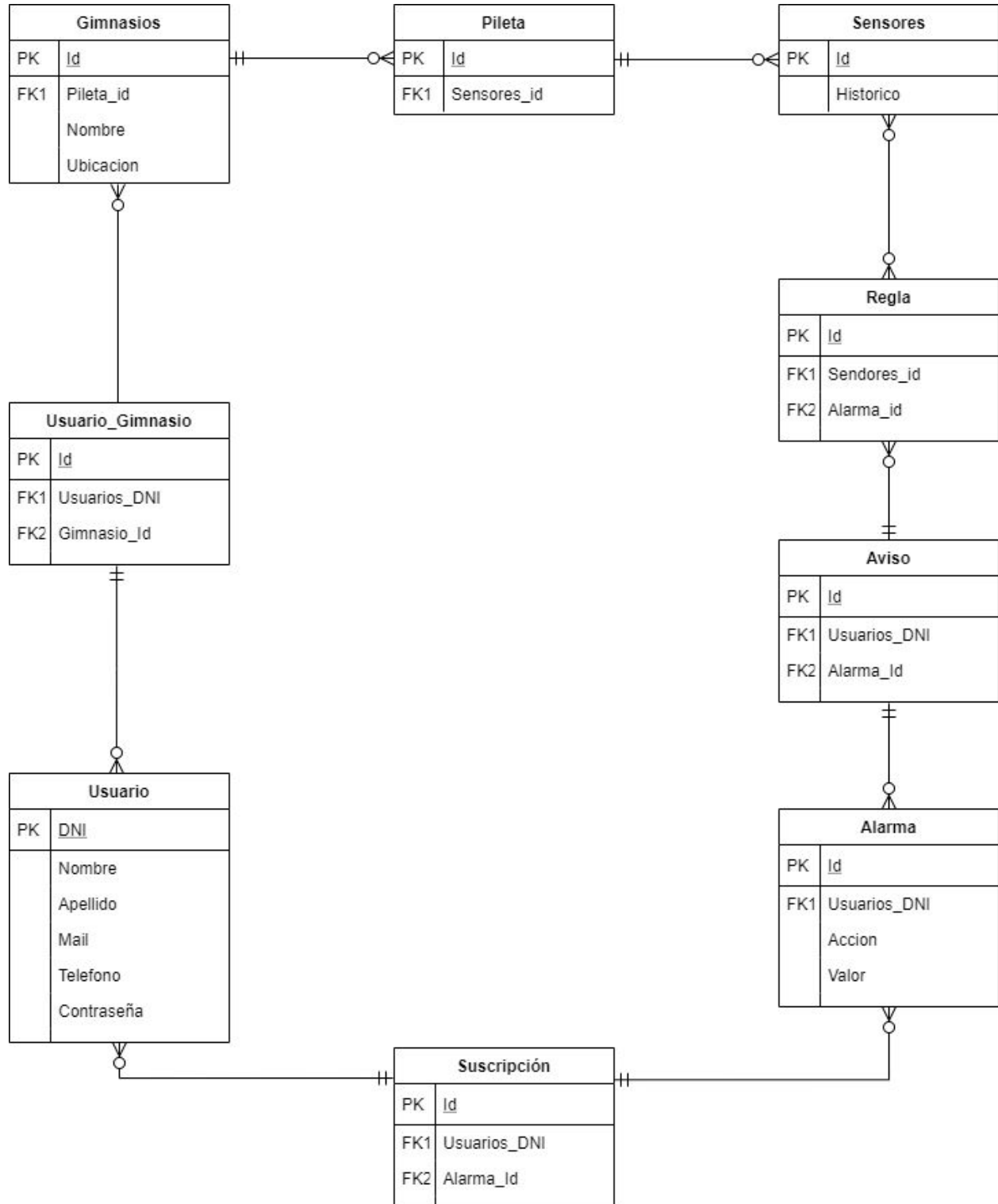
Lo hacemos posible.

Diagrama Entidad-Relación



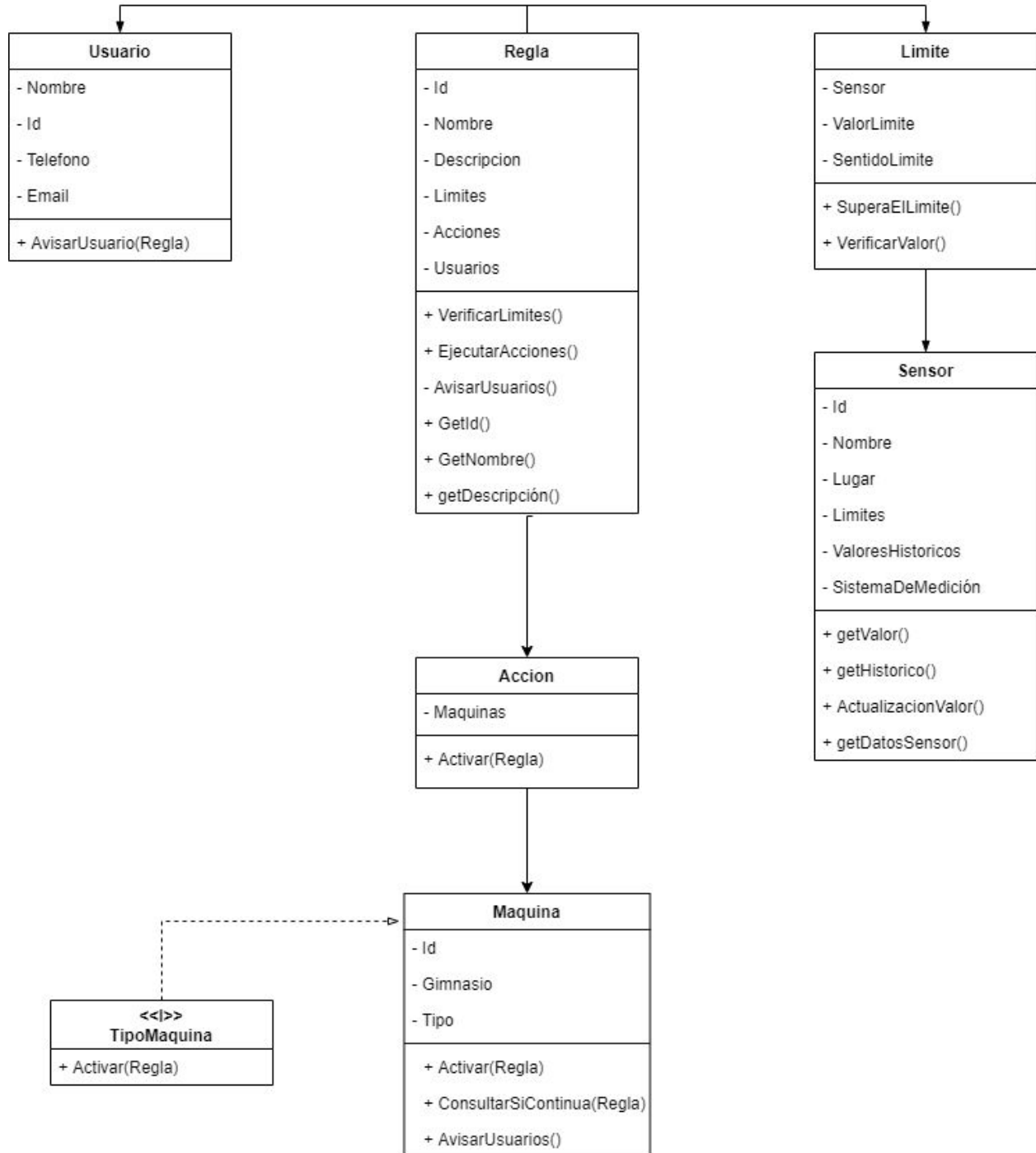
Lo hacemos posible.

Modelo de Datos Relacional



Lo hacemos posible.

Diagrama de clases



Lo hacemos posible.

En el diagrama se puede apreciar el funcionamiento de una alarma de la siguiente manera. Para conectar un sensor el objeto sensor debe subscribirse a este. Cada vez se actualice el valor se ejecuta `actualizarValor()` que envía el nuevo valor a los límites que miran ese sensor. Si el límite no es sobrepasado todo queda igual. En caso contrario la regla deberá detectarlo ese acontecimiento en la función `VerificarLimites()`. Cuando una regla es alertada esta revisa el resto de límites para ver si los demás también fueron sobrepasados mediante la función `superaElLimite()` en los límites. En caso negativo nuevamente no sucede nada. En caso positivo esta regla ejecuta `avisarUsuarios()` y `ejecutarAcciones()`. `AvisarUsuarios()` ejecuta en cada usuario que utilice la esté registrado en la regla la función `AvisarUsuario(Regla)` la cual envía al usuario los datos de la regla activada. En `EjecutarAcciones()` lo que se hace es pasar por cada acción de la regla y ejecutar `Activar(Regla)`, está activa en cada máquina que esté registrada en esa acción la función `Activar(Regla)` y cada máquina actuará de manera distinta. Las máquinas pueden utilizar la información de la regla para volverla a llamar para avisarle, una vez que terminaron, si la regla vuelve a estar activa mediante la función `continuarSiContinua(Regla)`. De esta forma es como funcionaria una alerta en el sistema

Maquetas de interfaz de usuario

Indice

1. Página de inicio
2. Página de login
3. Página principal
4. Página para la herramienta de gestión de gimnasios
5. Página para la herramienta de gestión de sensores
6. Página de los sensores de un gimnasio

Justificación

Este diseño busca reducir al mínimo los colores y detalles de la página a fin de no cansar la vista. Utiliza el blanco (#FFFFFF) como color de fondo y contraste para no cansar a los ojos; un azul claro (#42A5F5) como principal para resaltar la relación del club con la natación y su color complementario naranja (#F59142) para resaltar botones y notificaciones. Todos los bordes son simples y cortantes. En la tipografía se usa Segoe UI en Semibold con tamaños 60 para títulos y botones secundarios y 37 para títulos secundarios. Segoe UI en regular con tamaño 35 para botones secundarios e información y en 12 con cursiva para el botón de recuperación de contraseña.

Para mas detalles del diseño se recomienda ver “Intefaz S.A.P.: Modelos de diseños Básicos” en la siguiente url:

Lo hacemos posible.

https://docs.google.com/document/d/1jyzkFJ9rgEqauyLM53bSgyvjp1Hrcb1sqR_jrSRp0Dk/edit?usp=sharing

Maquetas

- 1) Página de inicio

CLUB Y GIMNASIOS DEVOTO sports

Entrar

- 2) Página de login

DEVOTO sports

Accede con tu cuenta

Usuario

Contraseña

¿Olvidaste tu contraseña?



Ingresar

Lo hacemos posible.

3) Página principal



4) Página para la herramienta de gestión de gimnasios

Gimnasios				
Codigo	Nombre	Gerente	Localizacion	Opciones
01	Constituyentes	Martin Fierros		✎ Editar ✕ Eliminar
02	Saavedra	Roberto Guerrero		✎ Editar ✕ Eliminar
03	Villa Urquiza	Jose Maria Domingo		✎ Editar ✕ Eliminar
04	Villa Devoto 1	Maria Jose Sabado		✎ Editar ✕ Eliminar
05	Villa devoto 2	Manuela Izquierda		✎ Editar ✕ Eliminar
06	Palermo Norte	Joseph David Sarmiento		✎ Editar ✕ Eliminar
07	Palermo Ashe	Tuvi ejagorda		✎ Editar ✕ Eliminar

Lo hacemos posible.


5) Página para la herramienta de gestión de sensores


DEVOTO sports

Herramientas de gestion
Gimnasios
Usuarios
Sensores
Reglas
Visualizacion
Gimnasios
Sensores
Alarmas


Gimnasios				
Codigo	Nombre	Gerente	Localizacion	Opciones
01	Constituyentes	Martin Fierros		↑ Seleccionar
02	Saavedra	Sebastian Hierros		↑ Seleccionar
03	Villa Urquiza	Jose Maria Domingos		↑ Seleccionar
04	Villa Devoto 1	Maria Jose Sabados		↑ Seleccionar
05	Villa Devoto 2	Manuela Izquierda		↑ Seleccionar
06	Palermo Norte	Joseph David Sarmiento		↑ Seleccionar
07	Palermo Ashe	Tobias Herjagorda		↑ Seleccionar

6) Página de los sensores del gimnasio


DEVOTO sports

Herramientas de gestion
Gimnasios
Usuarios
Sensores
Reglas
Visualizacion
Gimnasios
Sensores
Alarmas

Sensores	
Temperatura 1	14C°
Temperatura 2	13C°
Temperatura 3	16C°
Temperatura 4	17C°
Sensor PH 1	20ph



Listado de Componentes

- Servidor central para el registro de usuarios
- Servidores locales para la recolección de información de los sensores y la aplicación de las reglas
- Routers para la conexión de red
- Switches para conectar las computadoras de los gimnasios
- Sensores necesarios
- Arduino y/o Raspberry pi

Protocolos de Comunicación

- MQTT con el framework Mosquitto para la conexión de los sensores
- HTTP para la utilización de las APIs de los sensores
- HTTPS para el envío de datos sobre los usuarios
- TCP/IP para la comunicación entre los servidores locales y el servidor central

Diagrama de Arquitectura de la Solución del Alto Nivel

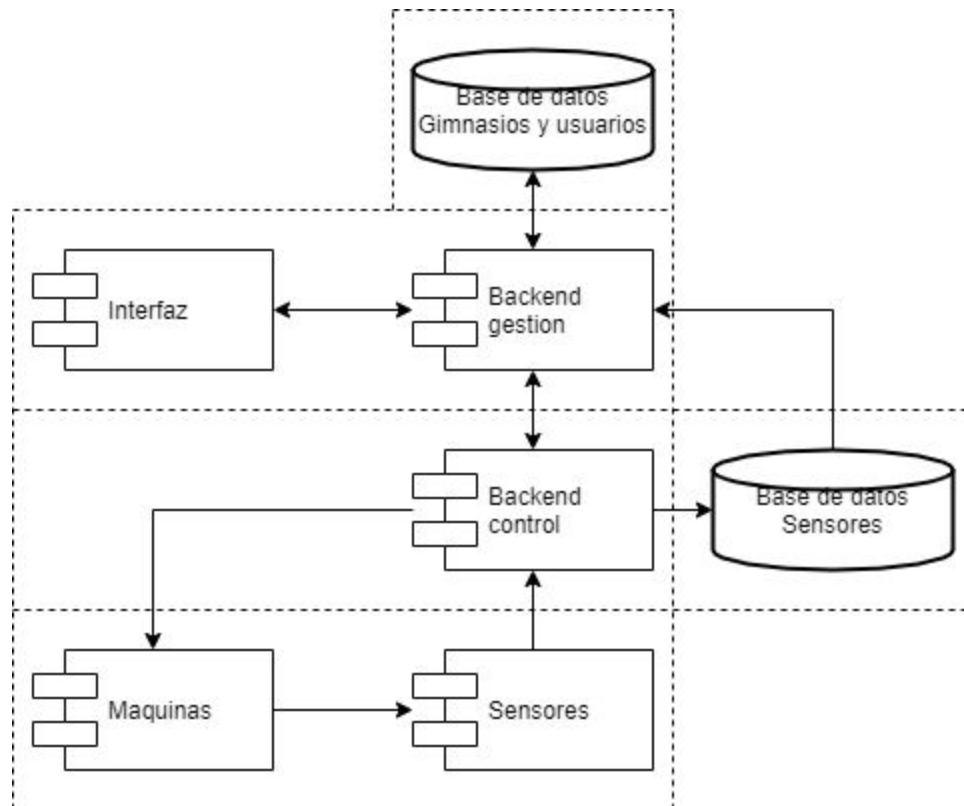
Para empezar primero intentaremos identificar son los drivers de calidad que requiere el sistema. En este sistema es prioritario la fiabilidad, la funcionalidad y eficiencia. La fiabilidad es necesaria para evitar errores graves como dejar salir mas cloro del necesario, utilizar agua caliente del termotanque cuando este no tiene agua o no está a la temperatura requerida o en caso contrario que se gasten recursos utilizando gas cuando esté disponible el agua del termotanque. Es por eso que se decidió utilizar dos sistemas separados; uno para el control y otro para la gestión de gimnasios y usuarios. La funcionalidad se requiere debido a la seguridad en el acceso ya que un cambio en las reglas o un robo de información pueden ser extremadamente perjudiciales para todos los involucrados. A esto se le suma que se requiere una exactitud bastante certera a la hora de hacer las mediciones y dejar a la máquina tomar decisiones. Por último la eficiencia se debe al hecho de que se están utilizando sensores esparcidos por todos los complejos que recolectan datos, envían resultados, reciben órdenes y ejecutan acciones. A pesar de estar dentro de un ambiente controlado la ineficiencia energética, temporal e de información pueden ocasionar problemas como una utilización excesiva de ancho de banda para controlar y verificar todos los sensores y máquinas, un uso excesivo de energía para mantener el sistema y una ralentización debido a la cantidad de datos que requieren utilizar la red.

En base a estos criterios se decidió utilizar dos sistemas separados e independientes el uno del otro pero que se comuniquen de forma constante aunque únicamente en momentos necesarios. De esa forma la seguridad de uno no debería afectar la del otro, se reducirían los

Lo hacemos posible.

recursos necesarios en los controles y los usuarios pueden desentenderse del control directo de los sensores y máquinas y de esa forma y como resultado extra se obtiene una buena usabilidad sacrificando cierto grado mínimo de mantenibilidad.

He aquí el diagrama que muestra ambos sistemas con bases de datos y backends separados pero con plena comunicación el uno con el otro.



Programa en java

Descargar el git: <https://github.com/producodes/GonzalezDesarrolloITAHK> y utilizar la carpeta Gimnasios