**Capstone “Malt technology”**

Arquitectura de software

línea horizontal

**Integrantes:**  **Docente:** Cindy Contador

Humberto Letelier **Asignatura:** Capstone

Álvaro Farias **Sección:** 009D

Sebastián Brenet

### 

# 

|  |  |  |
| --- | --- | --- |
| **VERSIÓN** | **FECHA** | **AUTOR** |
| 1.0 | 22-09-2024 | Sebastián Brenet |
| **RAZÓN** | | |
| 1.0 Creación del documento | | |

# Índice

[**Índice**](#_heading=h.9bjvxvjzwr7h) **2**

[1. **Introducción**](#_heading=) 3

[**2. Visión general de la arquitectura**](#_heading=) **3**

[**3. Descripción de los componentes del sistema**](#_heading=) **4**

[**4. Diagrama de arquitectura**](#_heading=) **5**

[**5. Integraciones externas**](#_heading=) **5**

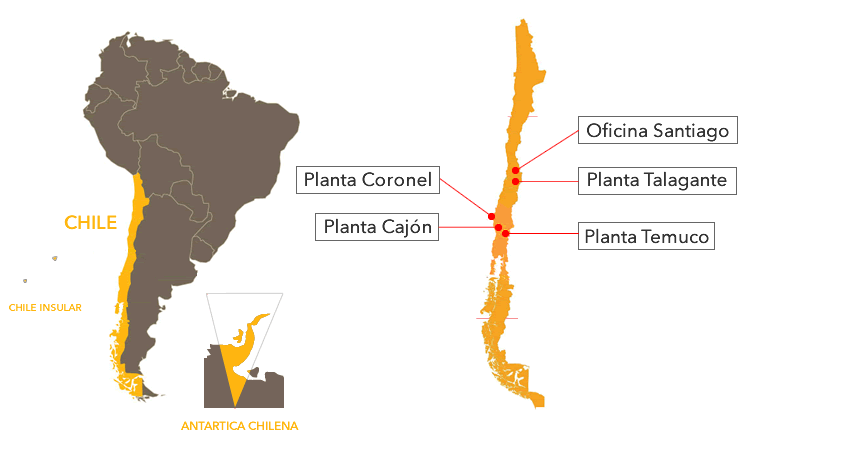
[**6. Patrones de diseño utilizados**](#_heading=) **5**

[**7. Escalabilidad y rendimiento**](#_heading=) **6**

[**8. Seguridad**](#_heading=) **6**

[**9. Mantenimiento y actualización**](#_heading=) **6**

[**10. Conclusión**](#_heading=) **7**



### 1. Introducción

**Propósito del documento**:

Este documento describe la arquitectura de software utilizada en la aplicación móvil de Maltexco, enfocada en la automatización de los procesos de captura de datos de las máquinas y su visualización en tiempo real. El propósito es optimizar el trabajo manual actual y proporcionar a los operarios una herramienta digital para gestionar los datos de producción.

**Alcance**:

La aplicación cubrirá la extracción de datos (como temperatura y humedad) de tres máquinas, su limpieza y almacenamiento en una base de datos, permitiendo la consulta en tiempo real mediante una aplicación móvil. La arquitectura también incluye mecanismos de seguridad, rendimiento y escalabilidad para garantizar la confiabilidad del sistema.

# 2. Visión general de la arquitectura

**Arquitectura Propuesta:**

La solución sigue un modelo cliente-servidor, diseñado para garantizar la interacción eficiente entre el cliente (aplicación móvil) y el servidor, que administra los datos de las máquinas.

**Cliente:**

* La aplicación móvil permite a los usuarios (operarios) registrar parámetros y visualizar los datos de las máquinas en tiempo real.
* Está desarrollada con tecnologías como Ionic Angular para garantizar una experiencia de usuario responsiva y multiplataforma.
* Utiliza llamadas a una API REST para intercambiar datos con el servidor.

**Servidor:**

* Se encarga de extraer, procesar, limpiar y almacenar los datos de tres máquinas distintas.
* Los datos se almacenan en una base de datos estructurada, como PostgreSQL o Oracle SQL, optimizada para consultas rápidas desde la aplicación móvil.
* El backend está desarrollado en Django, que expone endpoints RESTful para la interacción con el cliente.

**Arquitectura Distributiva:**

La arquitectura distribuye las responsabilidades entre los componentes del sistema:

* El cliente móvil se centra en la experiencia de usuario y la recolección de datos.
* El servidor realiza tareas intensivas, como procesamiento de datos, limpieza y almacenamiento.
* Esta distribución reduce la carga en el cliente y asegura que los datos estén siempre disponibles y actualizados.

**Arquitectura de Capas:**

La solución se organiza en varias capas para garantizar modularidad, escalabilidad y mantenibilidad:

**1. Capa de presentación (Frontend):**

* Compuesta por la aplicación móvil.
* Gestiona la interacción del usuario y la presentación de datos.
* Implementa validaciones ligeras y comunicación con el backend a través de la API.

**2. Capa de negocio (Backend):**

* Procesa y gestiona la lógica de la aplicación.
* Realiza tareas como la validación avanzada, procesamiento de datos y generación de alertas.
* Expone endpoints RESTFUL para que el cliente pueda interactuar con los datos.

**3. Capa de datos (Base de datos):**

* Administra el almacenamiento persistente de los parámetros de las máquinas.
* Almacena datos históricos y en tiempo real para consultas y reportes.

Esta arquitectura asegura una separación clara de responsabilidades, mejorando la escalabilidad y facilitando el mantenimiento futuro del sistema.

**Componentes principales**:

**Frontend (Aplicación móvil)**: Se utilizarán tecnologías como **Ionic** y **Angular** para crear y visualizar los datos en la aplicación.

**Backend (API y lógica de negocio)**: Responsable de la extracción, procesamiento y almacenamiento de los datos.

**Base de datos**: Base de datos relacional **PostgreSQL** para almacenar los datos estructurados.

**Integraciones**: API que conecta el backend con las máquinas y sistemas externos.

**Seguridad**: Implementación de mecanismos de autenticación y autorización, cifrado de datos en tránsito y en reposo.

# 3. Descripción de los componentes del sistema

**Módulo de frontend:**

**Funciones y responsabilidades**: La aplicación móvil permite a los operarios visualizar en tiempo real los parámetros de las máquinas, como temperatura y humedad. También podrán recibir alertas cuando algún parámetro exceda el rango establecido.

**Tecnologías utilizadas**: Angular – ionic ( typescript – css – html) para desarrollar la aplicación móvil. Se comunicaría con el backend mediante APIs **REST**

**Módulo de backend:**

**Funciones y responsabilidades**: El backend es el responsable de extraer los datos de las máquinas, realizar procesos de limpieza, almacenarlos y servirlos a la aplicación móvil. También es responsable de la lógica de negocio y la autenticación.

**Comunicación con la base de datos**: La API del backend realiza consultas a la base de datos para almacenar y recuperar los datos.

**Módulo de base de datos:**

**Estructura de las tablas y relaciones**: La base de datos almacenará las lecturas de temperatura, humedad y otros parámetros capturados por las máquinas. Más adelante incluirá un registro histórico para análisis y consulta a largo plazo.

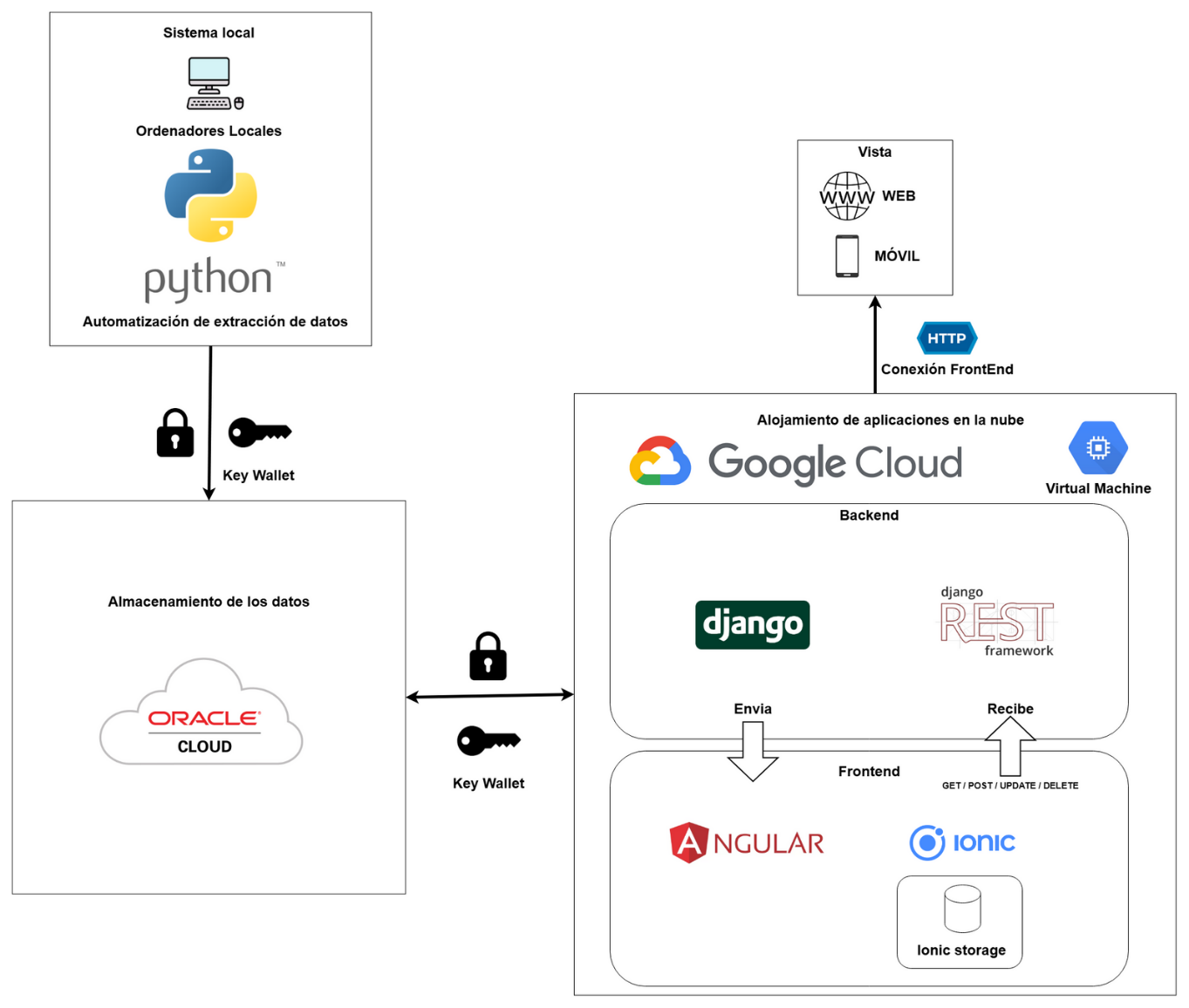
**Gestión de transacciones y consultas**: Las transacciones de datos estarán optimizadas para mantener la integridad de los datos y asegurar la consistencia en su acceso.

**Seguridad y backups**: Se implementarán mecanismos de seguridad para proteger los datos sensibles y realizar backups regulares.

# 4. Diagrama de arquitectura

**Proceso General:**

1. **Recepción de archivos .dat**: Recepción de los archivos de las máquinas.
2. **Lectura y procesamiento**: Implementar un sistema para abrir esos archivos, interpretar su contenido y extraer la información clave.
3. **Almacenamiento**: Los datos procesados se almacenarán en una base de datos, probablemente estructurada para que la aplicación móvil pueda acceder a ellos fácilmente.
4. **Visualización**: Los datos serán visualizados en la app para que los operarios y otros roles puedan revisarlos en tiempo real o mediante informes.



# 5. Integraciones externas

**Servicios :**

**Se usarán servicios básicos para enviar notificaciones a los operarios cuando haya problemas como un grupo de WhatsApp.**

**APIs:**

**El sistema tendrá APIs para que la aplicación móvil pueda conectarse, y estas serán seguras, usando un sistema de autenticación.**

# 6. Patrones de diseño utilizados

**Patrones de arquitectura**:

La arquitectura de backend estará basada en **microservicios**, lo que permitirá escalar y modularizar el sistema fácilmente. La aplicación móvil seguirá el patrón **MVC (Model-View-Controller)** para separar la lógica de la interfaz de usuario.

**Módulos reutilizables**:

Se diseñarán módulos para la autenticación, la extracción de datos y la gestión de errores, lo que facilitará su reutilización en otras partes del sistema.

# 7. Escalabilidad y rendimiento

**Mecanismos de escalabilidad**: El backend y la base de datos estarán alojados en un único servidor, ya que el sistema será utilizado por un máximo de personas en una primera etapa. Esto permitirá una infraestructura simple y eficiente. En el futuro, si es necesario, se podrá escalar horizontalmente añadiendo más servidores .

**Optimización de rendimiento**: Se utilizarán índices en la base de datos para mejorar la eficiencia de las consultas desde el inicio.

# 8. Seguridad

**Autenticación y autorización**: Se usará un sistema sencillo como  **(JSON Web Tokens)** para asegurar que solo los usuarios que hayan iniciado sesión puedan acceder al sistema.

**Cifrado de datos**: Toda la comunicación entre la aplicación móvil y el servidor estará protegida usando **HTTPS** para que los datos viajen de manera segura. Además, los datos sensibles dentro de la base de datos estarán cifrados.

**Medidas de seguridad**: Se tomarán precauciones básicas para evitar ataques comunes, como evitar que los usuarios inyecten código malicioso o accedan a datos que no les pertenecen.

# 9. Mantenimiento y actualización

**Estrategias de mantenimiento**:

Se implementarán sistemas de monitoreo para supervisar el rendimiento y detectar problemas. Además, se realizarán backups automáticos y se contará con procedimientos de recuperación ante fallos.

**Versionamiento del software**:

El control de versiones se gestionará mediante **Git**, y se implementará un flujo de integración continua (CI/CD) para asegurar que las actualizaciones se implementen de manera fluida y sin afectar la operación.

# 10. Conclusión

La arquitectura propuesta para Maltexco automatiza el proceso de captura de datos de las máquinas y permite su visualización en tiempo real mediante una aplicación móvil.Con el sistema, se espera mejorar la eficiencia operativa y reducir el riesgo de errores humanos. Además, el enfoque modular y escalable garantizará la flexibilidad y crecimiento futuro de la solución.