Sensor Data Platform Web Application

**Design Document**

**Project 3**

*Fall, 2013*

**Arie Radilla Laureano**

**Isil Demir**

**Lyman Cao**

**Yazhisai Gowthaman**

**Carnegie Mellon University**

**SILICON VALLEY**

Table of Contents

1 Background 3

2 Architectural Design 3

2.1 Metadata 3

3 Architectural Approaches 4

3.1 MVC 4

3.2 RESTful 4

3.3 In-memory database 5

3.4 Stateless 5

3.5 Non-blocking I/O 5

3.6 Layered 5

4 Design 5

5 Database Information 6

6 Implementation 6

7 Conclusion 7

# Background

Sensors have become a predominant part of digital interaction with the physical world. Managing and viewing these sensors is an integral part of accessing and using their data. In an attempt to simplify accessing this sensor data, we have come up with the Sensor Data Platform Web Application.

The application aims to provide a platform for the general public to access the sensor data. By means of this application, user can view sensors, devices they are attached to, their locations and the sensor data. In addition, users can also see a snapshot of all available sensors with accurate information of active and inactive sensors at any given time. Apart from providing a platform to view sensor related information, the application also acts as a channel for users to provide feedback to CMU.

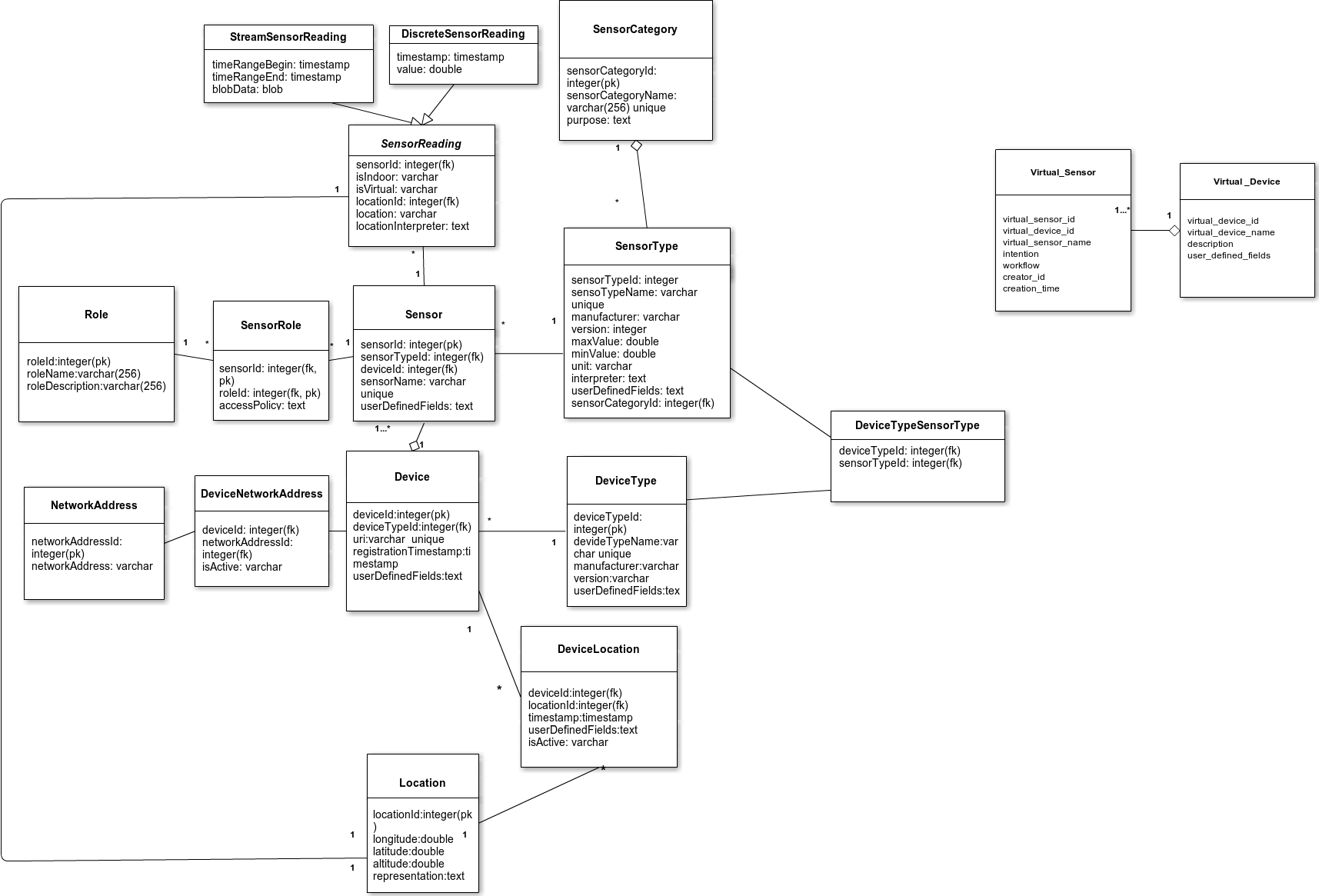
Our vision is to develop this web application with special focus to enable to it to be independent and extensible to handle higher loads (in terms of access) and open to a larger public as a one-stop point for relevant sensor data.

# Architectural Design

The architecture design of this project relies on the metadata to demonstrate the important components.

## Metadata

The metadata is the core of the application. All relevant data about the Sensors and the sensor data are represented in the database as the metadata. Below is a schema of the metadata for the application.

**Figure 1 - Metadata**

# Architectural Approaches

Listed in this section are the architectural approaches utilized in the Sensor Data Platform web site. They were picked and implemented for the primary reason of focusing on quality attributes such as modifiability, security, performance, extensibility and reliability.

## MVC

The Model-View-Controller paradigm allows the web application to be modular, and developed rapidly. The separation of the view and the business logic allows multiple view implementations for a single model, as well as offering a flexible development environment where frontend and backend concerns are separated. This separation also ensures that the application is more secure and extensible making the application open to future enhancements.

## RESTful

The web framework on which the website is built is entirely RESTful, which makes it very easy to scale the web application, as well as introducing the concept of caching client requests and greatly improves performance. REST’s separation of client and server is especially important in increasing effectiveness of performance tuning and scalability of server components.

It is also widely practiced in web development, thus bringing the web application up to par with industry standards.

## In-memory database

The database engine the web application uses for local data persistence can be configured as an in-memory database, which allows queries to run much faster, compared to tables stored on a disk.

## Stateless

The application also follows a stateless implementation, meaning that the client states are never stored in the server. Having each application instances to be completely stateless means that client requests can be load balanced across all the available virtual or physical machines. This allows a perfect venue for scalability and portability.

## Non-blocking I/O

The application’s web framework allows the server to handle client requests asynchronously. That is to say, a client request can be processed while other processes continue to run in parallel, making the web application highly efficient in handling concurrent client requests.

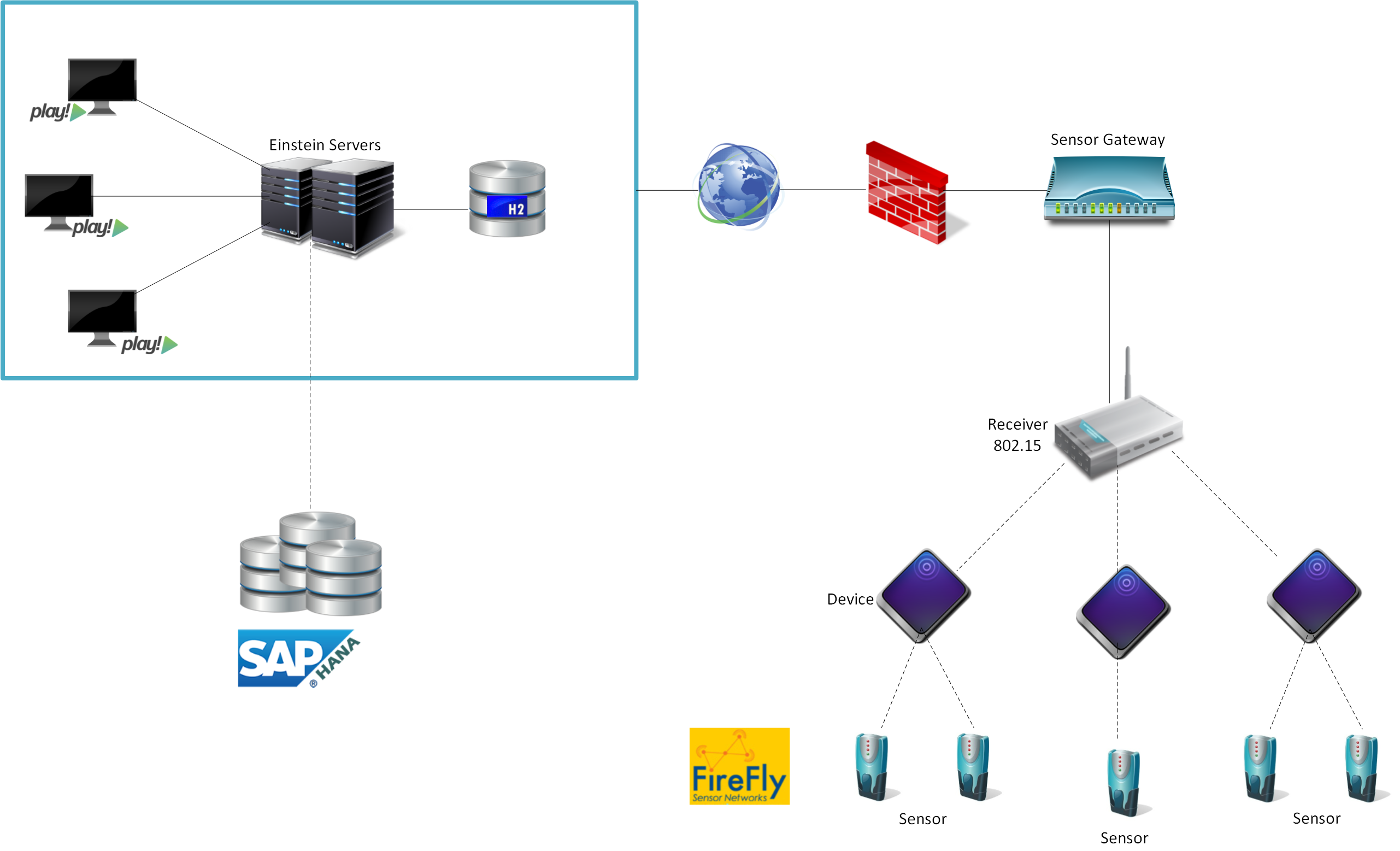
Considering that the application is expected to expand and be used by a large number of users, this improved the overall efficiency.

## Layered

The web application uses a completely separate set of API calls for its actual set of functionality. This allows the developers to easily differentiate between the web application and the API layer, in case of failures.

# Design

The following architecture diagram demonstrates the components and communication of the Sensor Data Platform Web Application.



# Database Information

In addition to the HANA database where the primary information about sensors and relevant sensor is stored, the web application also uses an additional database for local storage of application specific data. This data includes user information and bug reports.

The application provides “Sign In” functionality to enable authorized users to perform additional operations such as deleting and creating sensor components and viewing bug reports. Such user information is stored in a local database – the H2 database engine.

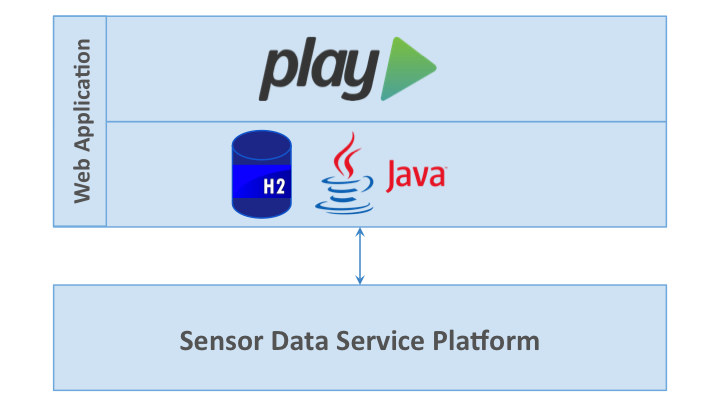
“Bug Reports” are also local to the application and is not a part of the Sensor Data Platform. This is also stored in the H2 database.

More information of H2 can be found at the below location.

<http://www.h2database.com/html/main.html>

# Implementation

The web application is implemented to communicate with the Sensor Data Platform as represented in the diagram below.



# Conclusion

The federated sensor data platform is a large scale cloud platform that will continue to grow via the integration of new sensor and user data. The primary goal of the web application is to make this data available to all end-users who wish to use it, without going through the necessary steps of implementing the API.

The concerns this project tried to handle are majorly focused on points like performance and scalability, as well as modifiability, due to the quickly changing nature of the API and sensor data.  Future extensions for this project should continue to regard scalability, performance and availability as main concerns. They should also keep the current architecture’s flexible nature to allow for growth, like making user-generated data an additional part of the platform.

The current web application is perfectly equipped to handle future complexities, while answering immediate user needs in a simple and efficient manner. The feasibility of the architecture and design can be further tested and revealed as more users start using the web application.