Fourth Wall Requirements Document

Group 26 - Finbourne Technology

*Daniel Whelan*

*Adam Mulvihill*

*Xiaolei Zhang*

*Daniel Madaghjian*

*Tamunotonye Harold Karibiye*

*Haojun Xing*

Table of Contents:

[1. Introduction 2](#_Toc95921826)

[1.1 Overview - Purpose of the System 2](#_Toc95921827)

[1.2 Scope 2](#_Toc95921828)

[1.3 Objectives and Success Criteria 2](#_Toc95921829)

[1.4 Definitions, Abbreviations 3](#_Toc95921830)

[1.5 References 3](#_Toc95921831)

[2. Current System 3](#_Toc95921832)

[3. Proposed System 4](#_Toc95921833)

[3.1 Overview 4](#_Toc95921834)

[3.2 Functional Requirements 5](#_Toc95921835)

[3.3 Non-Functional Requirements 5](#_Toc95921836)

[3.4 System Prototype (Models) 6](#_Toc95921837)

[3.4.1 Use Cases 6](#_Toc95921838)

[3.4.2 Object Model 8](#_Toc95921839)

[3.4.3 Dynamic Model 9](#_Toc95921840)

[Client Sign-off 10](#_Toc95921841)

# 

# 1. Introduction

## 1.1 Overview - Purpose of the System

The purpose of the system is to allow users to gain access to statistics related to the database such as queries, transactions, and index usage, as well as to retrieve information from certain configured system tables, without gaining access to client data present in the database. The purpose of this is to increase security, as the current procedure to access these statistics and queries is to request elevated access permissions, which gives full access to the database including client data.

Team Objectives

* Develop a service that will run inside a database cluster in parallel with each instance of a database to analyse them.
* Develop an authenticated REST API that will allow access to the service.
* Use PostgreSQL and C# for development.

## 1.2 Scope

We have been tasked with creating a service to analyse a database for table statistics and index usage. The service also should have the ability to keep track of the number of running queries and open transactions, and the ability to run a given query through an explain analyse clause. This service will run inside the cluster that the database is in. The service must be accessible through an authenticated REST API.

## 1.3 Objectives and Success Criteria

The main objective of this project is to create a monitoring service. It will run in parallel with each instance of the Postgres database. By the end of the project, we should:

* Finish the monitoring service that can:

1. poll the database for table statistics and index usage regularly
2. track long-running queries and open transactions
3. run read-only queries and return the response under “explain analyse”

* Provide a separate database or an AWS S3 bucket storing the above information
* Provide a suitably authenticated REST API that could access the database
* Deliver a repository of working source code

## 1.4 Definitions, Abbreviations

API - Application Programming Interface - An interface or communication protocol between different parts of a computer program

REST - REpresentational State Transfer - An architectural style for providing standards between computer systems on the web, making it easier for systems to communicate with each other

AWS S3 - Amazon Web Service Simple Storage Service - An object storage service that offers industry-leading scalability, data availability, security, and performance

## 1.5 References

Finbourne Technology (2022), “*Project 8\_Finbourne”*, Retrieved via CSU33013 Blackboard page.

# 2. Current System

The team at Finbourne Technology are frequently required to inspect and possibly make changes to the state of Postgres databases in the environments for several reasons, such as:

* To identify long-running queries / transactions that may be impacting throughput.
* To analyse the performance of existing queries to identify improvements.
* To view statistics and index usage on one or more tables as part of upgrade preparation.
* To run transformer processes as part of data migrations or to concurrently create indexes.
* To retrieve information from certain configured system tables (i.e., no client data) (Finbourne, 2022).

Each of these activities require the use of a “Break-Glass” procedure to connect to the database in the Kubernetes cluster. The frequency of use, volume, and duration of these “break-glass” procedures is of concern to Finbourne as they should be reserved for use in situations that operate outside of Business-as-Usual.

Diagram

Description automatically generated

Diagram from Finbourne highlighting how the current system operates

# 3. Proposed System

## 3.1 Overview

The proposed solution is to develop a monitoring service that will be spun up in parallel with each instance of a Postgres Database. This service will poll the database for statistics related to the database tables and index usage, whilst also keeping track of open transactions and long running queries. This information will then be stored in a separate database, or possibly an AWS S3 bucket, and will be accessed by a suitably authenticated REST API. The service will also allow for the execution of queries (read-only) and return the generated response from the database under “explain analyse”.

Diagram

Description automatically generated

Diagram from Finbourne highlighting how they would like the proposed system to operate.

## 3.2 Functional Requirements

* Develop a monitoring service that polls databases for statistics related to tables and index usage.
* Include the ability to track open transactions and long running queries.
* Include ability to run queries (read-only) through use of “explain analyse”
* Store information collected in a separate database.
* Provide access to this database via a REST API that is suitably authenticated
* Provide sufficient testing of software to ensure security.

## 

## 3.3 Non-Functional Requirements

* Write software using DotNet (C#)
* Provide more metrics than those proposed to store in database
* Create a graphical user interface for REST API
* Create testing to ensure that there are no data leaks in transfer from database to API
* Important for the project to have a cool name.

# 3.4 System Prototype (Models)

### 3.4.1 Use Cases

Diagram

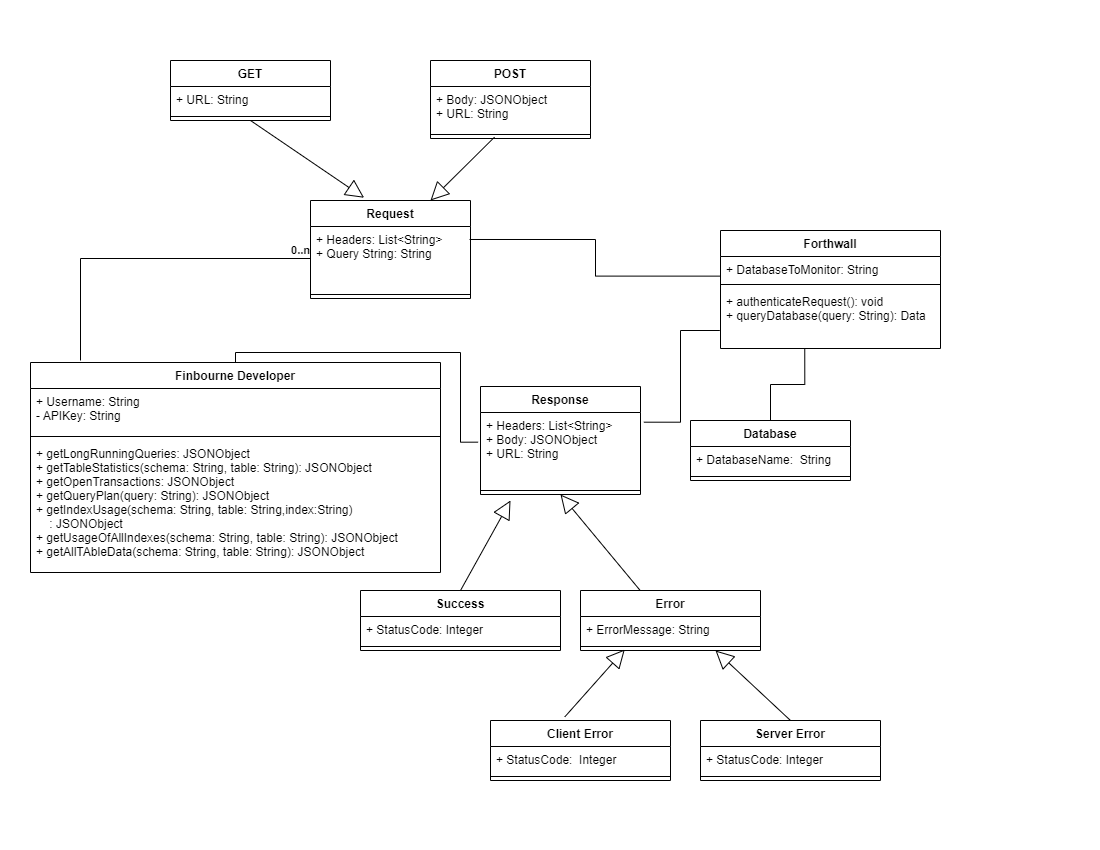
Description automatically generated

|  |  |
| --- | --- |
| **Title:** | Finbourne Platform Team processes API Request |
| **Summary:** | The Finbourne Team executes either a GET request or a POST request, which is then processed to the monitoring system (Fourth Wall) |
| **Actors:** | Finbourne Platform Team (Primary), Fourth Wall (Secondary) |
| **Preconditions:** | Finbourne Team executes the request correctly for it to be processed by the monitoring system. Finbourne Team know the Uniform Resource Identifier (URI) of the server. |

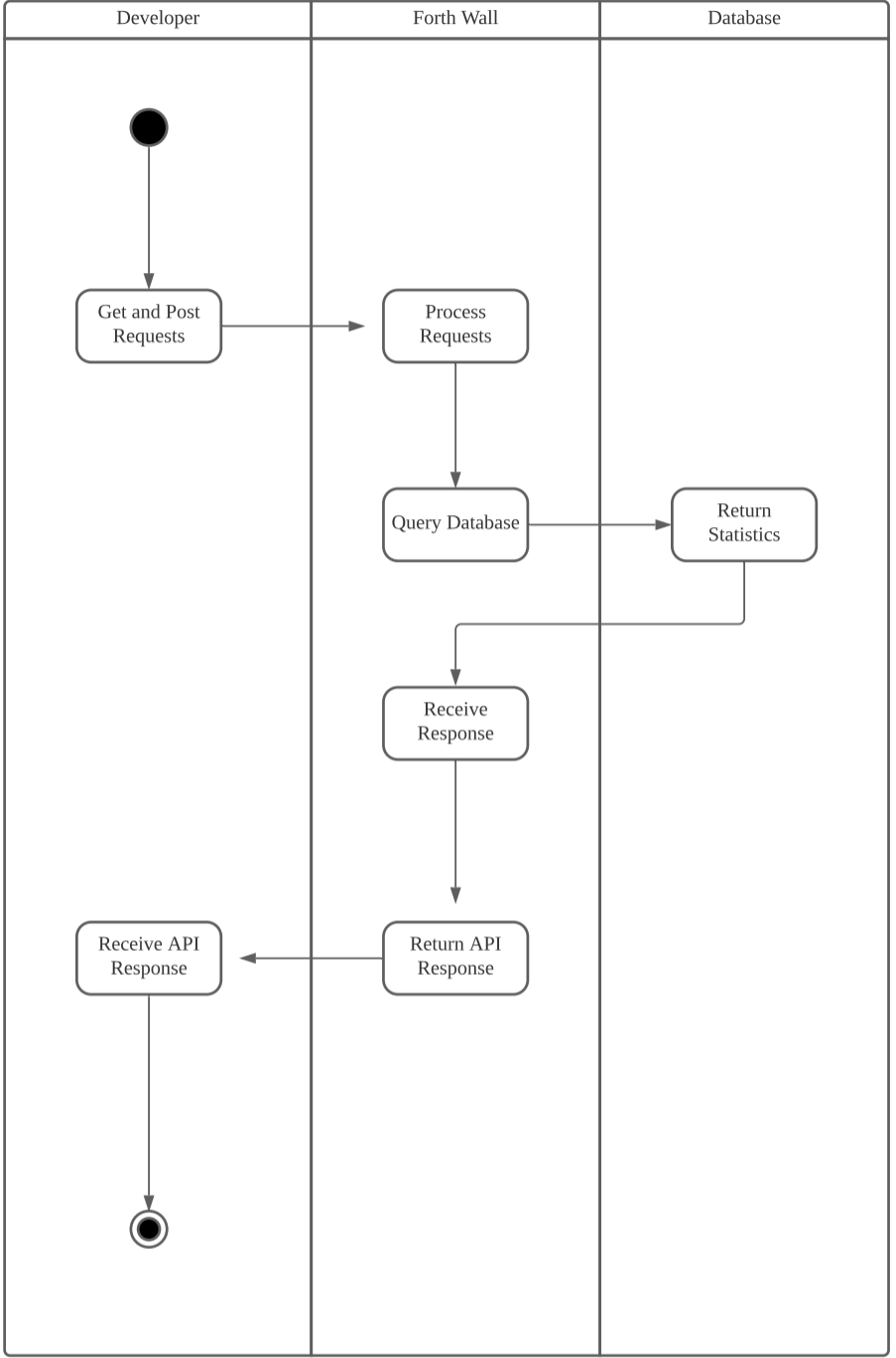
|  |  |
| --- | --- |
| **Title:** | Fourth Wall retrieves requested Data |
| **Summary:** | Once the monitoring system receives and processes the request from the team, they then investigate the database without break glass, to retrieve the requested metadata. |
| **Actors:** | Fourth Wall (Primary), PostgreSQL Database (Secondary) |
| **Preconditions:** | Monitoring system can inspect and retrieving the data without full access to the database. |

|  |  |
| --- | --- |
| **Title:** | Fourth Wall returns API Response |
| **Summary:** | Once the monitoring system retrieves the data requested, they then return an API response back to the Finbourne Team. This can contain a success status code or a client or server error code |
| **Actors:** | Fourth Wall (Primary), Finbourne Platform Team (Secondary) |
| **Preconditions:** | Monitoring system can inspect and retrieving the data without full access to the database. |

### 3.4.2 Object Model



### 3.4.3 Dynamic Model



# 

# Client Sign-off

