

Technical Report - **Product specification**

# KeyUsageProfiler

Course: IES - Introdução à Engenharia de Software

Date: Aveiro, 26-11-2023

Students: 107449: Miguel da Silva Pinto  
108287: Miguel Belchior Figueiredo  
108636: João Pedro Duarte Dourado  
110056: Ricardo Manuel Quintaneiro Almeida

Project abstract: Monitoring of key stroke presses and gathering of statistics around them

## Table of contents:

### [1 Introduction](#)

### [2 Product concept](#)

[Vision statement](#)

[Personas](#)

[Product requirements \(User stories\)](#)

### [3 Architecture notebook](#)

[Key requirements and constraints](#)

[Architectural view](#)

[Module interactions](#)

# 1 Introduction

KeyUsageProfiler is a web-application that allows a user to build teams and gather statistics about their team's keyboard usage. KeyUsageProfiler aims to simplify the supervision of various developers in a software development context by using a keylogger that sends to the team leader not only the key inputs of each member but also notifications about the inactivity of his members.

In the following chapter we'll go into more depth into the development of the product specification by analyzing use cases, creating personas and main scenarios, and developing user stories. According to these requirements an architecture will also be chosen. The product will be implemented following collaborative agile work practices and using GitHub as a code repository and project management system.

# 2 Product concept

## Vision statement

Our project focuses on developing a keylogger that tracks the inputs of various keyboards and stores usage data for each user. This data is given to the user through a web page according to the user's authorization. If the user is a team leader, he will have access to all the team members' information. However, each team member will only have access to self-related data and a leaderboards table if the team leader decides so.

This system was idealized mainly to help users obtain statistics about their typing data, which can be done as a self-evaluation, comparison between peers or as a monitorization tool for a manager in a company. The last one is more business oriented, allowing a visualization of workers' performance.

## Use Cases

### Actors:

1. Registered User
2. Team Member
3. Team Leader

### Use Cases for Registered User:

1. **Create Team:** A Registered User can create a new team.
2. **Join Team:** A Registered User can join an existing team.
3. **Login:** A Registered User can log into their account.

### Use Cases for Team Member:

4. **Login:** A Team Member can log into their account.
5. **View User Profile:** A Team Member can view their own user profile.
6. **Leave Team:** A Team Member can leave the team they are a part of.
7. **View Team Leaderboard:** A Team Member can view the leaderboard of their team.

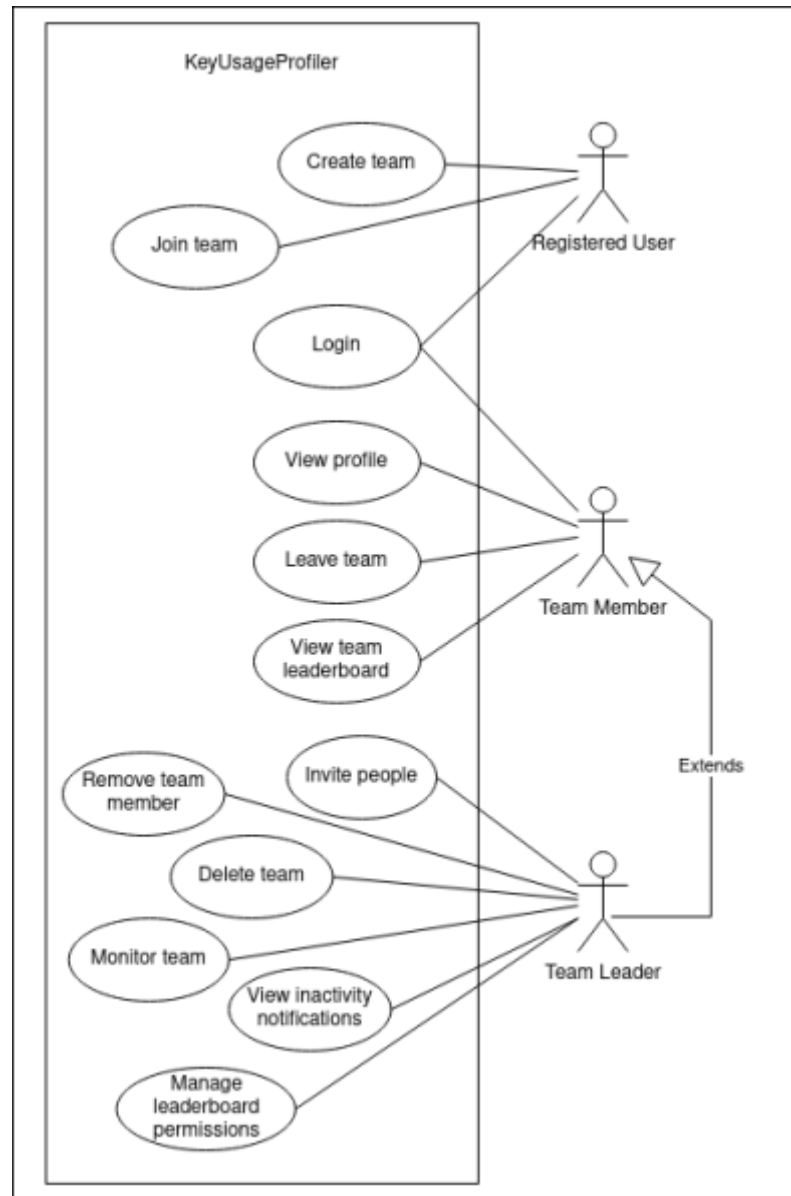


Image 1 - Use Case Diagram

### Use Cases for Team Leader (Extends Team Member):

8. **Invite People:** A Team Leader can invite other users to join their team.
9. **Remove Team Member:** A Team Leader can remove a team member from the team.
10. **Delete Team:** A Team Leader can delete the entire team.
11. **Monitor Team:** A Team Leader can monitor the team's activities and performance.
12. **View Inactivity Notifications:** A Team Leader can receive and then view notifications based on Team Member's lack of activity.
13. **Manage Team Leaderboard Permissions:** A Team Leader can manage permissions related to the team leaderboard.

## Personas

There have been defined two main actors for this system: **Team Leader** and **Team Member**.

**Edward (Team Leader)** – a 51-year-old, male, development team leader and SCRUM master.

- **Actor:** Team Leader
- **Motivation:** Wants to get a general and detailed view of the developers' performance in his team.
- **Requires** a service that gathers and presents relevant information about his overall team and each member when it comes to coding time per workday.



**Emily (Team Member)** – a 23-year-old, female, university student

- **Actor:** Team Member
- **Motivation:** Wants to compare her typing skills with her friends and compete with them.
- **Requires** a service that collects the typing data while she writes her essays and summarizes it while also allowing for a competitive view between the people in her friends group.



## Main Scenarios

### Monitoring team member's productivity

**Edward** has been noticing a decline in productivity in his team members and wants to monitor their productivity. So, in KeyUsageProfiler, he creates an account and subsequently a team, and invites his colleagues to the team to **see keypress statistics while at working hours**. He can also have a **more detailed view by checking certain patterns** (team members might be gaming at work) and **be notified of the user's inactivity**.

### Competing with peers

**Emily** is focusing on **competing with her friends to see who has the fastest typing speed**. To do that, she **receives an invite link for a team** in KeyUsageProfiler from one of her friends. Then she will install the program and run it with her credentials. After that, she will regularly **check the leaderboards** to see who is on top.

### Distributing workers in a project

**Edward** divided the team and distributed the tasks for the day. To guarantee that each group progresses at the desired rate he **watches each group's statistics separately**. Then he can allocate new members to the group that's underperforming.

### Helping team member's setup an environment

**Edward** is responsible for assuring that new members of a team can set up their environment to start development. With that in mind he checks for **team members that use the same IDE** as the new member and asks those members to help the newcomers with the setup process.

## Product requirements (User stories)

We identified 3 agile epics that encapsulate sets of related user stories - **user authentication, team management and statistics gathering and presentation**.

### User authentication:

User Story 1: [User login](#)

**As a** registered user

**I want to** log into my account on the website

**So that I can** see my user profile and the team I belong to

User Story 2: [User Registration](#)

**As an** unregistered user

**I want to** create an account in the website

**So that I can** access the website features

## Team Management:

User story 1: [Create Team](#)

**As a** registered user

**I want to** access the homepage

**So that I can** create a new team

User story 2: [Invite Team Members](#)

**As a** Team Leader

**I want to** create invite links

**So that I can** assign different people to a team

User story 3: [Become part of a Team](#)

**As a** registered user

**I want to** accept an invite link

**So that I can** become part of the corresponding team

User story 4: [Remove team member from team](#)

**As a** Team leader

**I want to** access a team management menu

**So that I can** remove a team member from the team

User story 5: [Leave current Team](#)

**As a** team member

**I want to** exit my current team

**So that I can** join another team or create my own

User Story 6: [Delete Team](#)

**As a** Team Leader

**I want to** delete my team

**So that I can** join another team or create another one

## Statistics gathering and presentation:

User story 1: [Key Heatmap](#)

**As a** Team Leader

**I want to** select a development team member

**So that I can** monitor the developer's performance by tracking the average key presses and visualizing it with a heatmap

User story 2: [User Profile Statistics](#)

**As a** Team Member

**I want to** go to my profile page

**So that I can** view how much time I spent typing, my average type speed and my peak typing time

User story 3: [Activity Analysis](#)

**As a** Team Leader

**I want to** see if a team member is gaming or coding

**So that I can** analyze if my team members are working or not

User story 4: [Live Virtual Keyboard](#)

**As a** Team Leader

**I want to** access a live representation of what a Team Member is typing

**So that I can** monitor the developer's real time activity on the keyboard

User story 5: [Estimation of IDE or editor](#)

**As a** Team Leader

**I want to** identify what editor are my Team Member's using when they are coding

**So that I can** know the IDEs used by my team

User story 6: [Leaderboards](#)

**As a** Team Leader

**I want to** access a dedicated leaderboard page,

**So that I can** conveniently track and compare team members' average typing speed and other useful statistics

User story 7: [Leaderboard Permissions](#)

**As a** Team Leader

**I want to** access the leaderboard settings  
**So that I can** limit leaderboard visibility by team members

User story 8: [Homepage Members Selection](#)

**As a** Team Leader  
**I want to** select multiple team members  
**So that I can** monitor their information as a group

User story 9: [Reset Team Statistics](#)

**As a** Team Leader  
**I want to** reset the key stroke statistics of my team members  
**So that I can** re-evaluate performance in a new agile sprint

User story 10: [Reset Statistics Periodically](#)

**As a** Team Leader  
**I want to** reset the key stroke statistics of my team members periodically  
**So that I can** evaluate performance in each agile sprint

User story 11: [Inactivity Notification](#)

**As a** Team Leader  
**I want to** be notified when team members are inactive  
**So that I can** keep my team productive during work hours.



## 3 Architecture notebook

### Key requirements and constraints

*KeyUsageProfiler* is a **service with a web interface** that displays statistics about a team's keyboard usage. Some of these statistics like average typing speed and peak typing speed require processing and others like the virtual live keyboard can automatically be displayed without any need for it. With this in mind, it makes sense for the latter to be directly fed to the webapp while the first passes through the required processing channels. But there is a catch: the data behind all features is the same (the keystrokes).

**Implementation** regarding the keylogger should be straightforward (collecting keystroke data), except if the need comes to support several alphabets and/or keyboard layouts.

**Performance issues** are a major concern: since data will be generated at a rapid pace, from every keystroke of every team member, performance should be kept in mind during architecture design. It certainly does not make much sense to insert data in the database on a per-key basis, otherwise there will be a transaction made in the database for each key pressed.

Here are some key requirements and system constraints that have a significant bearing on the architecture of *KeyUsageProfiler*:

- User Authentication and Authorization must be correctly implemented, so that each user only has the necessary permissions to perform their actions and cannot access sensitive data from other users when that was not intended.
- To key log information, users must authenticate on startup of the keylogger. Their keypress data will be passed through a Message Queue, with information related to the pressed key, who pressed it and when it was pressed (timestamp).
- Live keyboards should update in real time with data from the Message Queue related to users pressing keys.
- Performance will be an issue if we insert every keystroke into the database, since it would result in a lot of writes. To prevent this, we should buffer keystrokes and insert them according to a predefined timeslot.
- Using such data, the system should be able to generate statistics, such as key heatmaps, patterns and present them to the users who have authorization to access it.
- After receiving and processing the data from the Message Queue, to be displayed at any time in the future it should be stored in a database.
- There should be a REST API to allow the data to be displayed in JSON format and to be requested by the webpage when it needs it. The API should also allow the webpage server to send data generated on the webpage.
- User inactivity will be detected using a cache database and Spring Boot. A key

will be associated with a user, and when it expires, it will generate an event that will be detected by the spring boot backend. Those inactivity notifications will be passed to the frontend through a queue and saved into the persistence database.

- To avoid filling up the database with unneeded data, when a team is deleted, or its statistics are cleared (both allowed to Team Leaders only) all keystroke data should be erased from the database.

## Architectural view

The architecture planned for the KeyUsageProfiler consists of:

### Keylogger

- Takes key press data from users and feeds the Keystroke Distributor with it.

### Keystroke Distributor

- Responsible for distributing the keypress data to components that want it, such as the Live Keyboard and the Keystroke Consumer.

### Notification Distributor

- Responsible for distributing notification data, such as alarms about user inactivity to components that want it, such as the Web interface.

### Spring Boot

- Keystroke Consumer: Receives keypress data from the Keystroke distributor and passes that information to the Keystroke Saver.
- Keystroke Saver: Uses the interface provided by the Persistence Service and the Cache Service in order to cache keystrokes or save them persistently to the database, respectively.
- Persistence Service: used to interact with the persistent database. Responsible for saving keystroke data and calculating and saving statistics about that data.
- Cache Service: used to interact with the cache database.
- REST API: Serves the frontend services with information about the system, such as user information, statistics and keystroke data.
- Alarm Aggregator: gets information from the cache database about user inactivity. Uses this information to send alarms to the Notification Distributor (to notify in real time) and saves these notifications in the persistence database.

### Databases

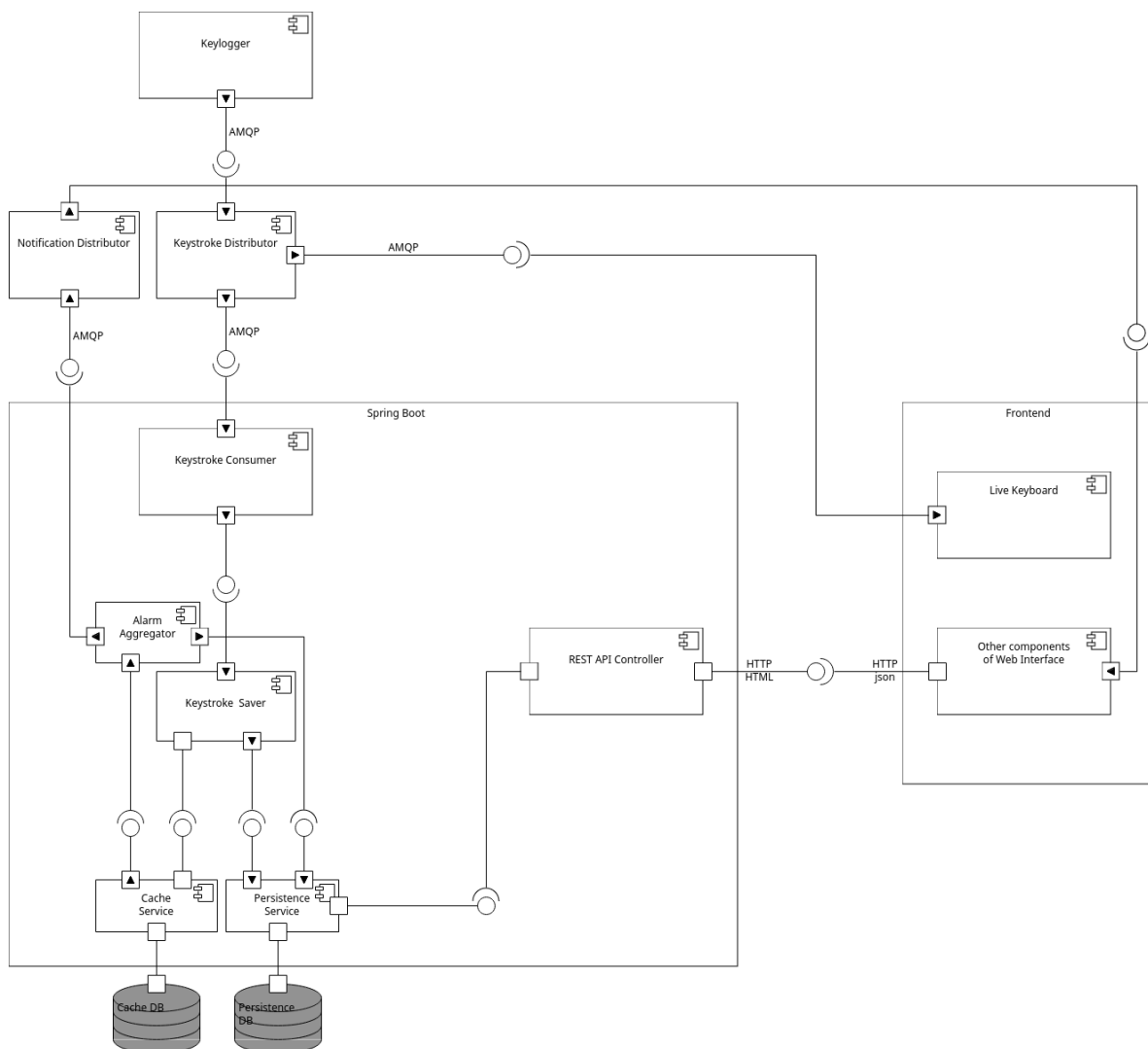
- Cache Database: Database used to store keystrokes being pressed and cache

those for a period of time, so that they can be then flushed to the persistence database. Used to detect user inactivity and “notify” the Alarm Distributor about it.

- Persistence Database: Database which will store all data that is meant to be permanent and upon which most queries will be made to show the key stroke statistics. Information saved includes: login information, notifications, statistics and teams.

## Frontend

- Live Keyboard: receives keystroke information from the Keystroke Distributor and shows them in real time on a graphical keyboard.
- Other components of the web interface: get desired information from the REST API and display it in the respective components.



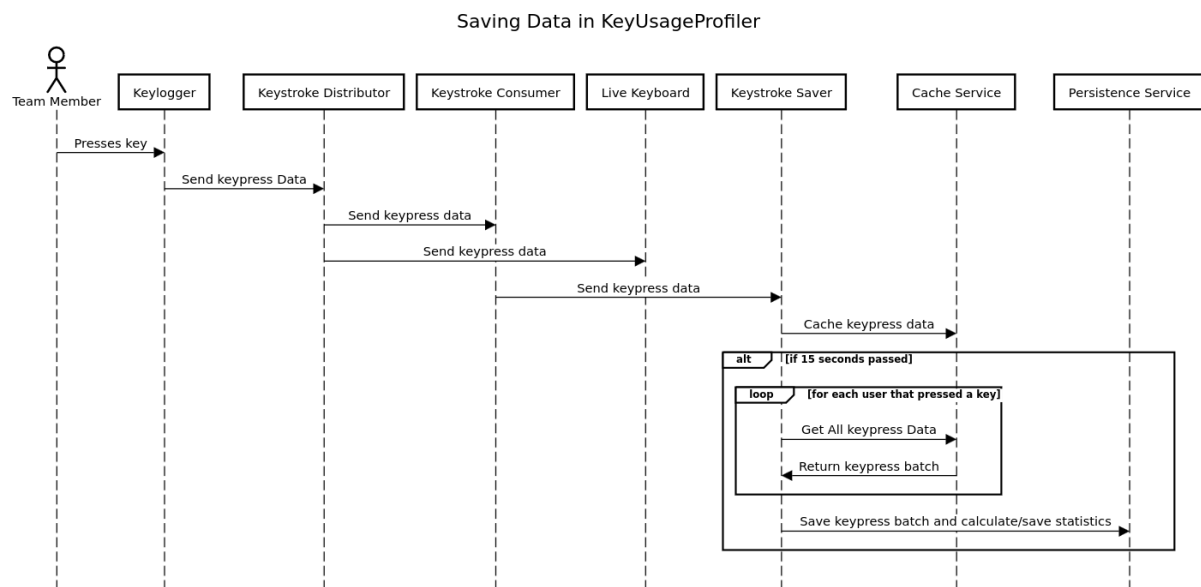


## Module interactions

We will split our module interactions into 2 main interactions:

### 1. Saving Data in *KeyUsageProfiler*

- A Team Member with the service running presses a key. That data is sent as an event to the Keystroke Distributor.
- The data is obtained by the Keystroke Distributor which sends it to both the Live Keyboard and to the Keystroke Consumer.
- The latter is responsible for forwarding the data to the Keystroke Saver which caches the data (through the Cache Service). Every 15 seconds, the keystroke Saver is also responsible to flush this data to the Persistence DB (through the Persistence Service) and calculate/save statistics related to the last sent keystrokes.



## 2. A Team Leader viewing a team member's statistics.

- The Team Leader accesses the profile of a team member through the dashboard or a link.
- The Web Interface as it loads the page contacts the REST API Controller and asks for the statistical information of the team member in question by using the team member's id.
- After receiving a request from the web interface, the REST API Controller requests the needed data from the Persistence Service.
- The Persistence Service sends the requested data to the REST API.
- The REST API Controller finally returns the statistical data in JSON format to the Web Interface.
- The Web Interface displays the team member data to the Team Leader.

Team leader viewing team member statistics

