Technical Report - **Product specification**

KeyUsageProfiler

Course:	IES - Introdução à Engenharia de Software
Date:	Aveiro, 03-11-23
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

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 webpage 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.

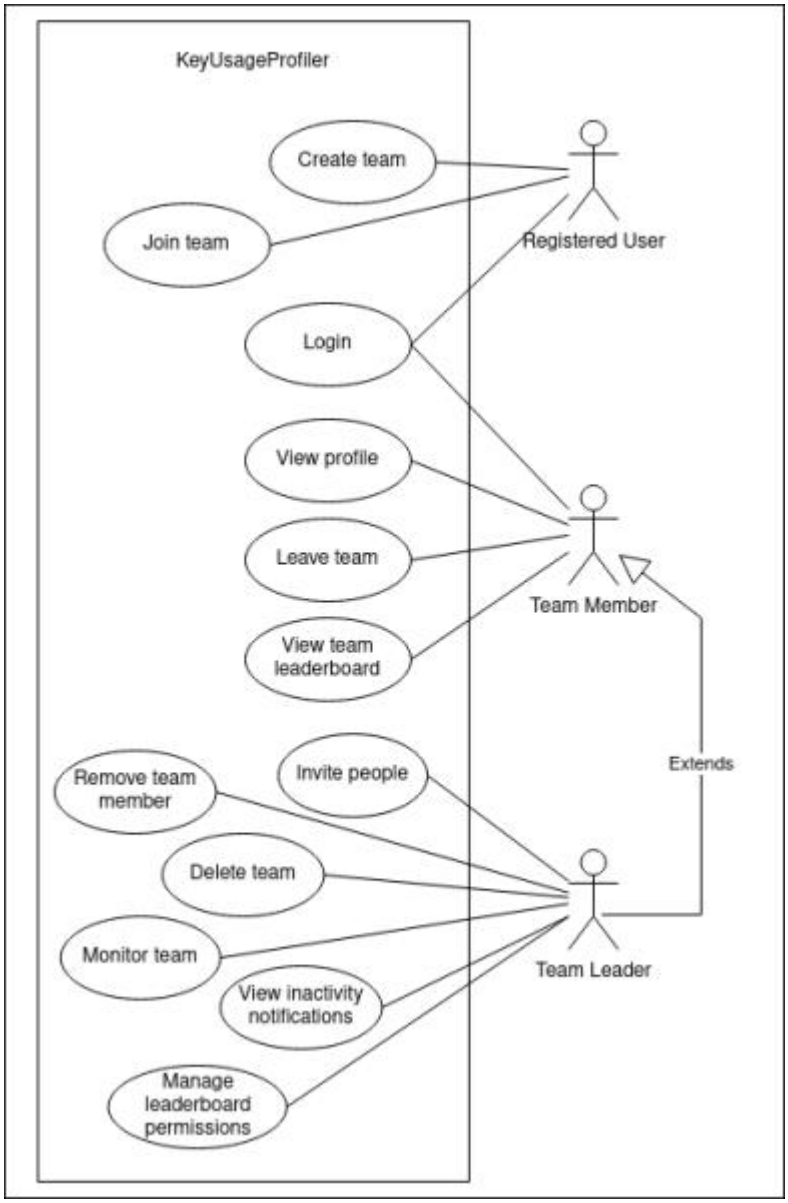


Image 1 - Use Case Diagram

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.

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 member might be gaming at work) and **be notified of 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 straight forward (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 once they reach a threshold. In the case that the buffer doesn't reach the threshold, there should be an operation to flush all keystrokes occasionally.
- 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 server when the webpage needs it. The API should also allow

the server to send it data generated on the webpage.

- In between the API and the web interface there should be a server that is meant to serve as middleware between them both, to hide the API from the public and also to serve as an intermediary between the Message Queue service and the front-end in real-time. As this server already handles real time information it will also be responsible for notifying the Team Leader of inactivity periods of the Team Members. However as we want these notifications to persist, the server will POST the Spring Boot API (in this case we'll interact with the API without it being a response to a user interaction).
- To avoid filling up the database with unneeded data, when a group is deleted, or its statistics are cleared (both allowed to Team Leaders only) all key stroke 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 event broker with it.

Event Broker (Kafka)

- Where the Message Queue will be created. Its messages will be served for consuming for the purposes of both the Live Keyboard and the processing and storing of the key stroke data.

Spring Boot

- Kafka Consumer: Consumes the messages in the Message Queue and forwards them to the Business Logic.
- Business Logic: Spring Boot will oversee processing data from the Message Queue and store it in the databases and/or make it available in the API.
- REST API: Makes information available to the Node.JS server which in turn it will pass it along to the client dashboards.
- MySQL Service & Redis Service: provide CRUD interface with the different databases.

Databases

- Redis: NoSQL Database used to store the current keystrokes being pressed. It will keep up to the last 64 keystrokes per user, which will be used to insert a batch of keystrokes into MySQL. To prevent not storing data because the user didn't get to 64 keystrokes, every 5 minutes the Redis keystroke data will be flushed in MySQL. Complex and slower queries to the MySQL database will be cached using Redis for the next 5 minutes.
- MySQL: Relational 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.

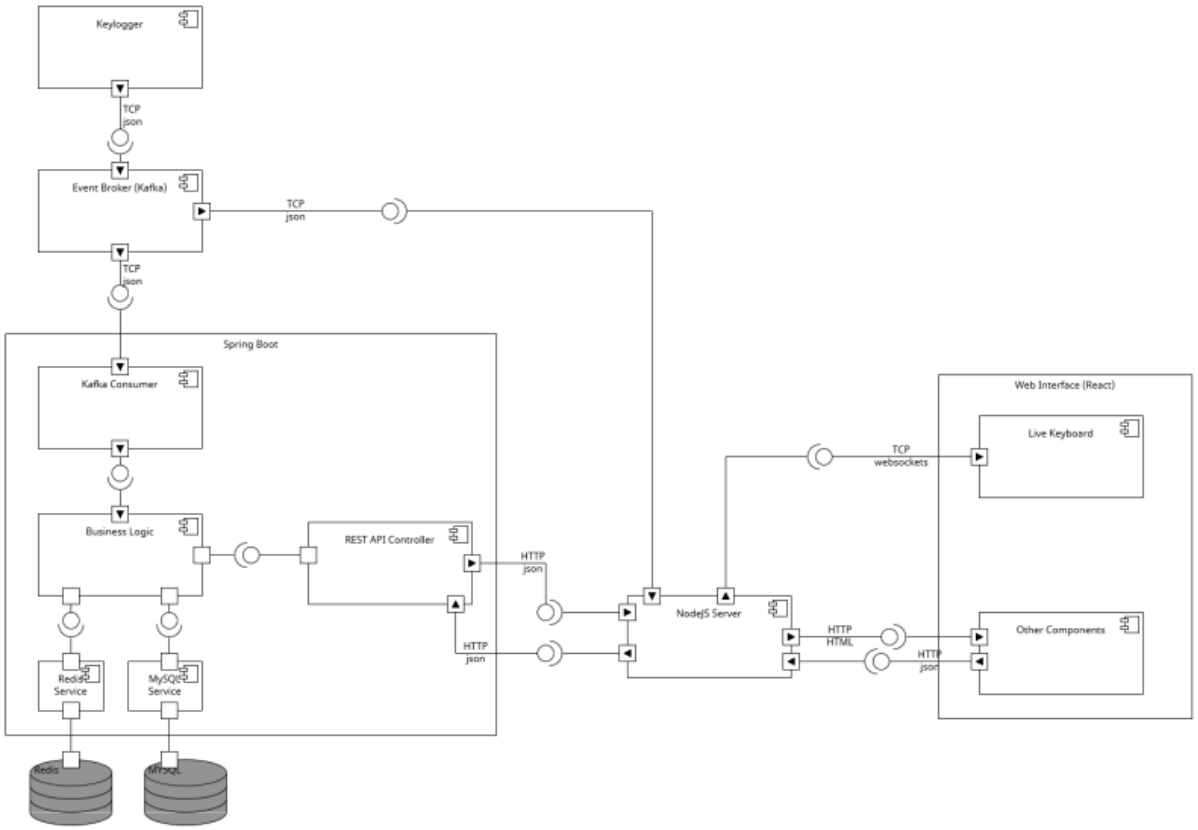
Information saved includes: login information, notifications, statistics, keystroke batches per user, teams.

NodeJS Server

- Will consume the REST API to fetch all the required data to the Web Interface, including the real-time (or close to it) communication needed for the Live Keyboard.
- Is responsible for processing raw data that comes from the keylogger with the goal of creating certain alerts for Team Leaders. Sends the notifications to the front-end and also to the API, through a POST request, to store the notification data in the databases, for history purposes.

Web Interface (React)

- The Web Interface will allow all our users to have access to the information that we intended, such as: sign up/log in; the ability to create, join, manage and delete a team (role dependent); see said team's statistics; see individual members statistics; leaderboards about certain statistics, among others.
- There is a webpage updating in real time: the live keyboard. It gets information through a websocket interface from the NodeJS Server and dynamically updates the keys that were pressed.

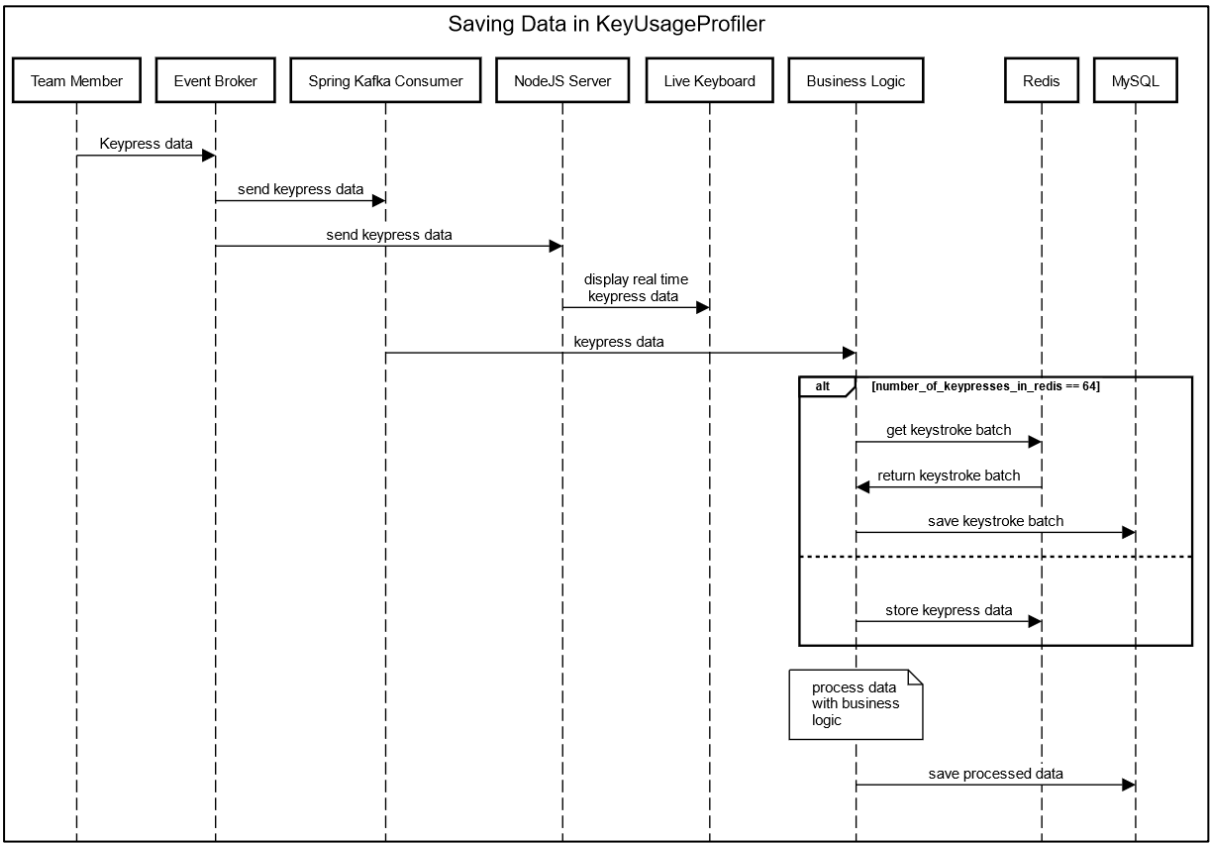


Module interactions

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 Event Broker (Kafka).
- The data is obtained by the Event Broker which sends it to both the Kafka Consumer (part of Spring Boot) and the consumer in the NodeJS Server.
- The latter (NodeJS server) allows the visualization of data in real time by sending it to the Live Keyboard component **through a Websockets interface** which makes the page update upon receival of new keys (which belongs to the Web Interface).
- The first one (Spring Boot consumer) forwards the data to the Business Logic where it will be processed and saved in both databases.
- The interaction with Redis is to batch key inputs and save them in MySQL once they reach a certain size.
- The interaction with MySQL is to save processed dat
- a, such as statistics (typing speed, number of typed words ...) and notification alerts.



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

- Team leader accesses the dashboard in the web interface.
- He is presented with the list of his team members and their status (coding, idling, etc.).
- He can view individual information about each of his team members by clicking the link on the list of users to their profile. This includes information such as: typing speed, a key heatmap and real-time updating live keyboard.
- The Node.JS server requests that data from the REST API and, after passing through business logic, that data is returned and rendered to the team leader for viewing.