Tehchical Report - SpeakByddy

Advanced Programming
Mariam Betin – 20232020300

Faculty of Engineering
Universidad Francisco Jose de Caldas

CONTENTS

Tehchical Report of SpeakByddy

Abstract— **Language learning has been a constant challenge due to the lack of practice with native speakers. SpeakBuddy is a platform designed to connect people who want to learn a language with native speakers through meaningful social interactions. A system based on language preferences and common interests was developed to facilitate user immersion in a new language. The results show a successful match rate and an improvement in participants' communication skills.**

I. Introduction

SpeakBuddy is an application designed to connect people interested in learning languages through interaction with native speakers. Its purpose is to provide a safe and dynamic space where users can improve their language skills through real-time conversations and intelligent pairings.

The application includes a registration form where users enter their personal information, the languages they master and those they wish to learn, as well as their level of proficiency in writing, reading, speaking and listening. In addition, personal interests are taken into account to improve matches between users.

II. USER STORIES

From a survey of different users, we found that they share interests and motives which can be coupled in the following stories.

- As a user, I want to find people with similar interests and language skills to enhance my learning.
- As a user, I want to be able to with people who share my interests, to have more natural and enriching conversations.
- As a user, I want to be able to modify my level of the language I am learning, so that the application can adjust the matches according to my current level.
- As a user, I want the application to ensure that my data privacy and interactions are secure, so that I can use it with confidence.

III. FUntional Requirements and funtional requirements

This may be a modified version of your proposal depending on previously carried out research or any feedback received.

A. Funtional Rqueriments

- User registration: Allow users to register and create profiles with their personal data, languages and interests.
- Smart Matching: Implement a system that connects users based on the languages they wish to learn and their common interests.
- Data storage: Store user information securely.
- Authentication and security: Manage user authentication and ensure data privacy.

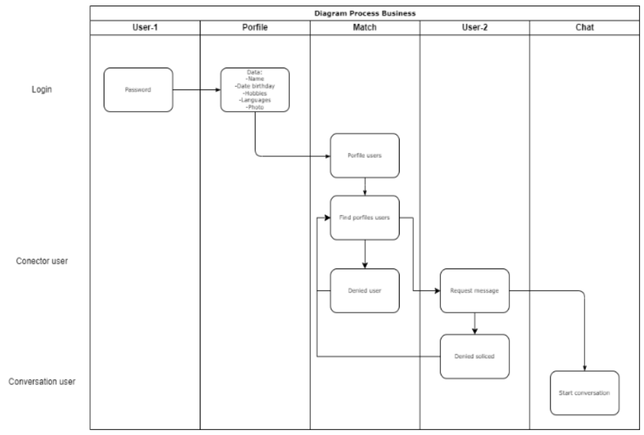
B. No Funtional Requirements

- Intuitive interface: The application must be easy to use, even for non-technical users.
- Scalability: The backend must be designed to allow for future enhancements and expansions.
- Performance: The system should respond in the shortest possible time for basic operations such as registration and matching.
- Privacy: User data must be protected and only accessible for authorized functions.

IV. UML diagrams

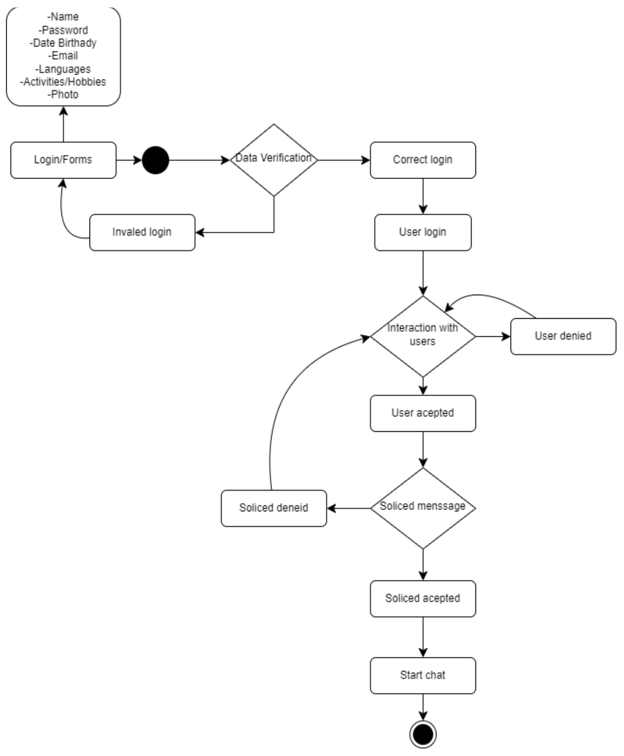
A. Secuence Diagram

This diagram shows the flow of interaction between the system components during the authentication and matching process. It starts with the user entering his credentials, which after being verified, are validated. Then, matching is performed based on the user's languages and interests, and finally, if both users are chosen, they are successfully matched.



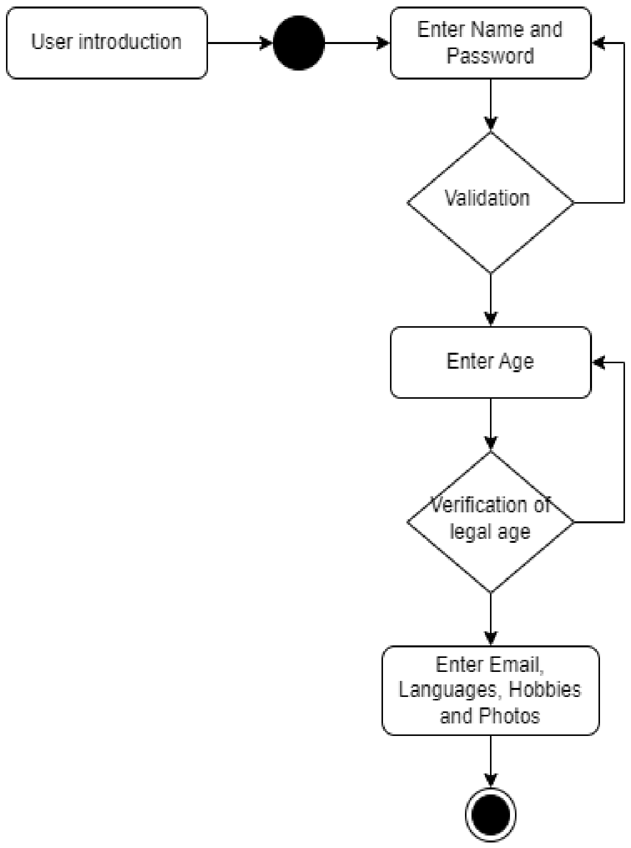
B. Activity Diagram

This diagram represents the flow of activities from user registration to matching. It includes steps such as profile creation, data validation, matching and display of suggested profiles. It is useful to understand how users can interact with the application and how their requests are processed.

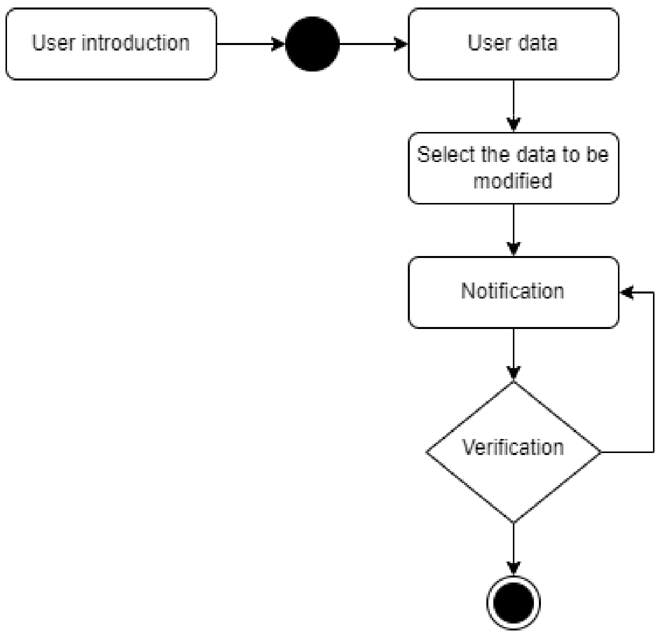


It presents the different interactions of the user with the application to try to be more precise in the expected actions, thinking about the most appropriate process.

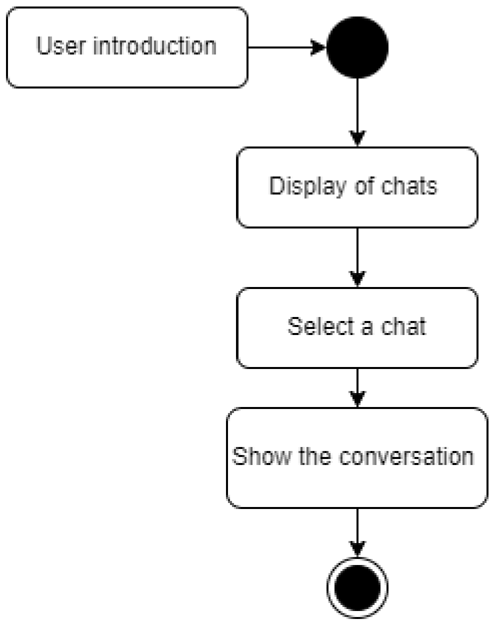
Form filling


User introduction
Enter Name and Password
Validation
Enter Age
Verification of legal age
Enter Email, Languages, Hobbies and Photos

Profile modification

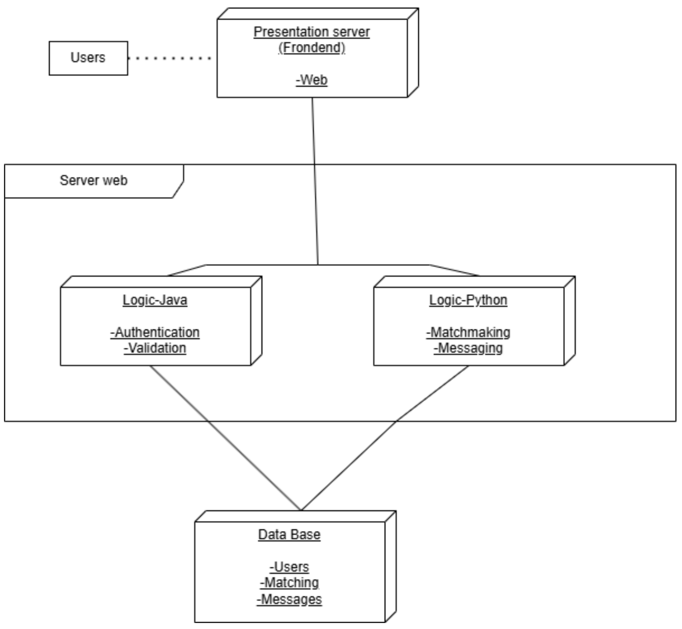


Interaction with chats



C. Depoyment Diagram

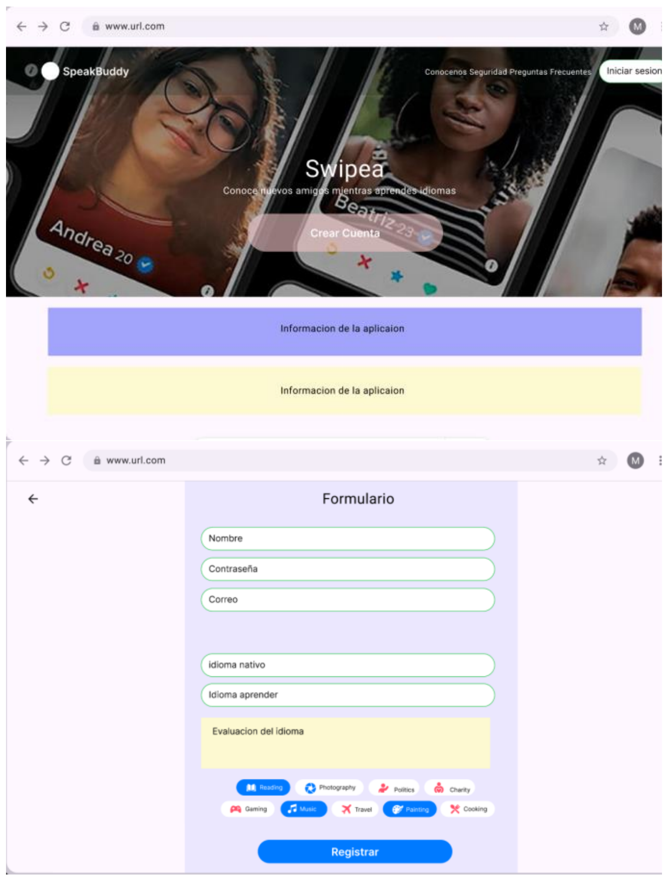
This diagram describes the system architecture, showing how the components are distributed between the frontend (Html, Css, JavaScript), backend (Java and Python) and data storage (JSON). It helps to visualize how the different parts of the system communicate to form the application

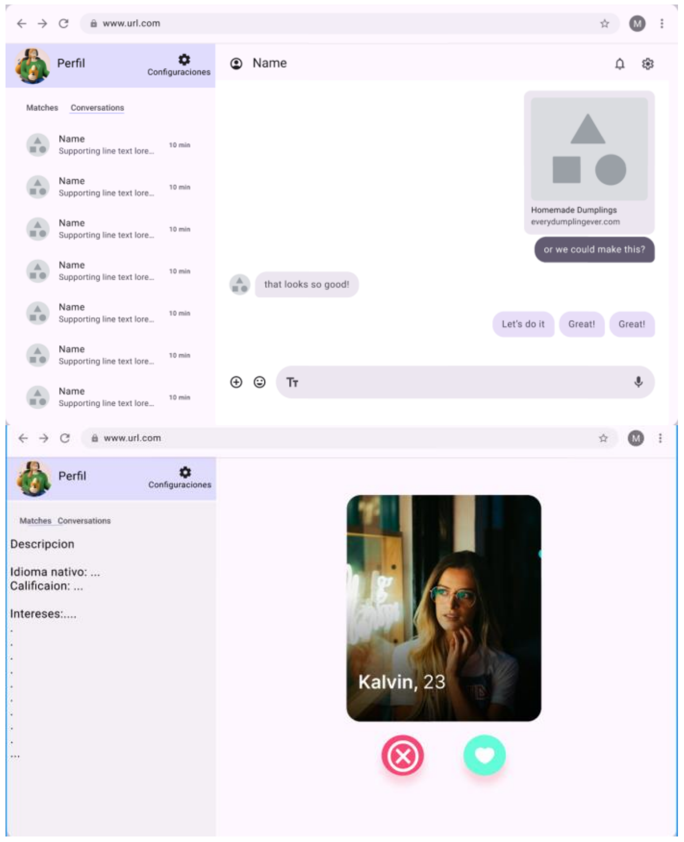


V. CONCEPTUAL DESING

A.Depoyment Diagram

The mockups show us an initial view of how the main screens of the application will look like, such as the application page, the registration view, the matching screen and the chat window. These sketches help visualize the user experience and ensure that the interface is clear and easy to use.





or we could make this?

that looks so good!

Let's do it

Great!

Great!

Descripcion

Idioma nativo: ...
Calificaion: ...

Intereses:....

B.CRC Cards

CRC cards support us in defining responsibilities of each class in the system.

user	
Responsibility	**collaborator**
Store personal information and language perference	Match UserRespository Message

UserRepo	
Responsibility	**collaborator**
Load and save user information	User

Match	
Responsibility	**collaborator**
Relate compatible user	User

MessageRepo	
Responsibility	**collaborator**
Save and retrieving messages	message

Message	
Responsibility	**collaborator**
Store and manage messages among user	User MessageRepo

VI. ARCHITECTURE

A. *Cliente-Server Model*

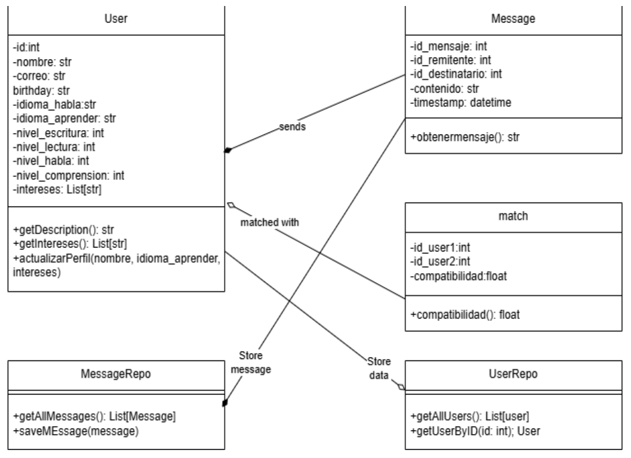
The backend is divided into two main components: Java and Python. The Java component handles RESTful API endpoints for user authentication, registration, and data validation. It ensures that user credentials are securely managed and validated before granting access to the application. On the other hand, the Python component is responsible for the matching logic and message management. Python's flexibility and powerful libraries make it ideal for implementing complex algorithms, such as the smart matching system, which pairs users based on language preferences and interests. Both components communicate seamlessly through JSON, ensuring efficient data exchange and scalability.

- Presentation Layer (Frontend): Developed with HTML, CSS and JavaScript, this layer is responsible for the user interface and user interaction.
- Application Layer (Backend): Divided into two parts: Java for authentication and data validation, and Python for matching logic and message management.
- Data Layer (JSON): Stores user information, preferences and messages sent.

This separation of concerns allows for easier maintenance and future enhancements. For instance, if the matching algorithm needs to be updated, it can be done independently of the authentication system, reducing the risk of introducing bugs or security vulnerabilities.

B. *Class Diagram*

This diagram defines the relationships between the main entities of the system, such as User, Matching and Messages. It helps to understand how the different classes interact with each other to achieve the functionality of the application.



VII. BACKEND

The backend architecture is divided into two main components: Java and Python. The Java component is responsible for handling RESTful API endpoints related to user authentication and data validation during login and registration. On the other hand, the python component manages the core logic for user matching based on language preferences and interests, as well as the storage and retrieval of messages. This separation of concerns ensures that each component focuses on a specific functionality, improving maintainability and scalability.

The use of Java for authentication ensures robust security practices, leveraging its strong ecosystem for handling sensitive user data. Python, known for its simplicity and efficiency in data processing, is ideal for implementing the matching algorithm and managing message exchanges. The communication between these components is facilitated through JSON, which serves as a lightweight data interchange format.

Some key services for the project are:

- /api/v1/users/auth (Java) → Verify user credentials.
- /api/v1/match (Python) → Return matches based on language and interests.
- /api/v1/messages/send (Python) → Saves sent messages in JSON.

VIII. D

IX. SOLID PRINCIPLES

The application follows SOLID principles to ensure robust and maintainable code:

Single Responsibility Principle (SRP): Each class in the system is designed to have a single responsibility, ensuring that changes in one part of the system do not affect the others.

Open/Closed Principle (OCP): The system is designed to be open to extensions, but closed to modifications. This means that new features, such as additional languages or improved matching criteria, can be added without altering the existing code base.

Liskov Substitution Principle (LSP): Derived classes can be used in place of their base classes without affecting the behavior of the system. This ensures that any extension or modification of the system is consistent with the existing architecture.

Interface Segregation Principle (ISP): Interfaces are designed to be specific to the needs of the classes that implement them, avoiding the creation of “fat” interfaces with unnecessary methods. This favors a cleaner and more modular code.

Dependency Inversion Principle (DIP): high-level modules (authentication and logic) are decoupled from low-level modules (such as data storage) by using JSON as an intermediary. This allows for greater flexibility and easier testing.

Following these principles, the application achieves a high degree of modularity and flexibility, making it easy to maintain and expand in the future.

X. CONCLUSIONS

SpeakBuddy offers an innovative and cost-effective solution for language learning by connecting users with native speakers in real time. The application's modular architecture ensures scalability, allowing for future enhancements such as additional languages or advanced matching algorithms.

The application's design prioritizes user experience and security, making it a reliable tool for language learners. By leveraging modern web technologies and adhering to software design best practices, SpeakBuddy is well positioned to grow and adapt to the changing needs of its users.

REFERENCES

[1] Universidad Distrital. (2025) Advanced Programming Lecture Notes. https://github.com/EngAndres/ud-public/tree/main/courses/advanced-programming/slides

[2] Duolingo press room. https://press.duolingo.com/#about

[3] Matching Algorithms in Dating Apps. https://arxiv.org/pdf/2006.06899

[4] Client-Server Model. https://www.geeksforgeeks.org/client-server-model/

[5] REST API documentation. What is Rest? https://restfulapi.net/

[6] Python Web Application. https://realpython.com/python-web-applications/