HELPIE, the app that helps mentally impaired people to travel alone

Technical Report

ATC 2024

P4

Jade Therras, NX
Claire Payoux, MTE
Eliser Romero, NX
Lena Florence De Sepibus, NX

September 6, 2024, Lausanne

Contents

1 Introduction

Travelling and mobility are essential in our society. Without them, we would not be able to discover new places, peoples and ideas. We would not be able to develop and construct society and life projects. As easy mobility is taken for granted by most of us - thanks to cars, bikes and public transportation, etc. - we do not always realize its importance. However for some people, mobility is difficult, which heavily impairs their independence.

In Switzerland, the SBB (Swiss Federal Railway) Mobile App is the most widely used application to help travelling through public transportation [1]. It features a lot of useful functionalities such as travel planning, real time information about schedule and location as well as ticket buying. However, this makes the application complex and inaccessible for certain people.

As the SBB are concerned by inclusiveness they have developed a second application: "SBB Inclusive" providing additional and adapted information for people with visual or auditory impairment.

In this project, we will concentrate on a different kind of condition: cognitive impairment. For individuals experiencing cognitive challenges, the complexity and high density of stimuli found in the public transportation system and SBB Mobile App can be overwhelming.

The Helpie App is an idea that was initially brought by *PlusSport*: an association that tries to gather people with all kinds of disabilities in sport events. They realized that one of the biggest challenges for people with cognitive impairment was the transportation of the different people towards the site of the event and that the SBB Mobile App was not adequately adapted. They therefore brought the concept of a simplified and adapted app to the Hackathon, a project that was made in collaboration with several railway companies like the SBB, but also the SNCF, OBB and the DB (respectively the french, austrian and german railway biggest representatives). Within a week-end, they developed some ideas and the mock-up of an app.

Helpie then eventually landed in our hands: the *Assistive Technology Challenge*, an EPFL-project whose primary goal is to develop a personalized solution for a person with disabilities. We were assigned a challenger that lies on the autistic spectrum, let's name him Charlie, and regularly met with him to adapt the progress of Helpie.

2 Methods

During this project, we worked with an iterative workflow, where the challenger was included in both the brainstorming and the testing part of each idea we could have.

This process first started when we met and travelled with Charlie to define his needs. Then, we kept brainstorming ideas and presented them to him through a *Figma* prototype and collected feedback until the result seemed satisfying. We then implemented the model into a functional *Android* app using *Kotlin* programming language and adapted it in function of technical constraints. Finally, we tested the prototype, first at rest for the design, and then in real travel conditions to evaluate its efficiency.

The choices of design were therefore directly taken from the user requirements that were defined during our discussions and experience with Charlie. Several challenges and goals were evoked during our discussions: the *SBB Mobile* app being too complex with too many information, the stress induced by the hyper-vigilance during a known trip, forgetting to take the ticket and the general disorientation due to the lack of adapted information (not knowing which bus is to be taken, at which schedule...).

3 Design

Our Challenger wanted an application with as few stimuli as possible, but with enough information to do a full journey. The following design choices were therefore taken in order to develop a simple app that would guide him through a journey:

A. **Global Theme** : We decided to keep the blue chosen by the first Hackahealth for the global theme. We aimed to create a clean and straightforward interface. Every element is designed to be easily understandable and accessible thanks to the color contrasts and the button shapes. In the **Appendix** section, several screens are displayed to show explicitly how the application looks like (see fig. 2).

B. **Start Screen** : The start screen is the main menu page. It includes settings and allows to start a travel

C. **Destination Screen** : The screen leads the user to the choice of destination. For starting the trip, a set of four pre-registrated relevant addresses are displayed, like "Home", "Work", or "Sport", so that he does not have to bother with entering the address. An additional feature would however permit to do so if needed.

D. **Journey Summary** : The journey summary is displayed with the estimated time of arrival and a description of each step. To follow on the journey, the user needs to click on the **START** button ("Commencer"). There is also a grey **Back** button ("retour") if one wants to change the destination, e.g. because of a precedent missclick.

E. **Take a ticket** : We have 2 possible interfaces to remind the user to take their ticket. The choice of this interface can be done using the settings. The first screen that we decided to implement is telling the user to take an "easy-ride" ticket using the official SBB application. By clicking on "Prendre le billet" the user will be transferred to the SBB app directly in the easy-ride section and only needs to drag the button to the right (explained on the screen). The ticket will automatically be taken and calculate its price by looking at the ongoing trip. However it happened that Charlie already had a ticket: in those situations we activate the other version of this interface where it reminds him to stamp his ticket.

F. **Ticket button**: The ticket button will be a easy way to access the ticket in the app, always accessible during the travel.

G. **Step by step instruction**: A step-by-step journey, where we try to detail the trip as much as possible while giving only the necessary information each time to not saturate the screen with too many stimuli. Therefore, all steps are given in a chronological order, one-by-one where the only possible interactions are to go forward through the trip or to interrupt the journey.

- For walk travels, for example to go to the next bus stop, *Google Maps* will be launched with the adequate destination. Charlie is quite used to deal with satellite maps, especially with *Google Maps* when he walks and requested for it.
- For public transport, essential information are clearly displayed such as the waiting time, the id, estimated time of arrival (ETA), direction of the transportation as well as the type of transportation.

H. **Appropriate notifications and warning**: Two types of notifications have been implemented in order to remind Charlie to go back to Helpie if he uses another app as well as giving him the appropriate reminders to go in/out a transportation for instance :

   - When leaving the app, a blue movable square will still be on the screen to remind the user of the ongoing travel and allowing them to go back easily to *Helpie*.
   - 2 minutes before the arrival, a yellow notification will pop up to warn the user to be ready for the change of transport. The same type of notification will pop up when arriving at destination of a check-point (e.g. the bus stop or train station)

I. **Help button**: A help button is always displayed, so that the user can easily call someone if they happen to be in a dire or stressful situation.

J. **Settings**: On the menu page, a setting button is available that will allow a close relative or the caregiver to personalise the user's experience. This button is for now accessible only on the start screen.

K. **Progress bar**: To help the user engage and use the app, a gamified progress bar was added to the display.

A typical journey will take this form :

- Choose the destination and take the ticket
- Walk to the stop (with Google Maps)
- Reach the stop, verification
- Wait for the transport, verification that you enter
- Wait in the transport, verification that you go out
- If another step is needed, then restart an iteration, otherwhise stop the ticket and end the journey

Our challenger speaks french, so the app has been implemented in French and in English for more flexibility.

4 Technical process

As the user is in possession of an Android phone, we decided to develop the app with Android Studio in Kotlin using JetPack Compose. The whole code is available on our git following this link: https://github.com/ATC-Helpie-2024 [2]. Please note that the following explanations concern the *Helpie 1.0* version, which is contained in the Git branch of the same name.

The code have been fully documented in KDoc, using DOKKA to generate the documentation. Run **./gradlew dokkaHtml** in the terminal of android studio to obtain the documentation, or decompress the documentation.zip file and open index.html. The documentation is also accessible in the git repository, in *documentation/index.md*.

Helpie had to communicate with several databases to be operational. First, the SBB API [3] was needed to access the trip request service and access not only schedules but also precise locations and check-point (steps) for more complex trips. This API needs to receive an .XML file and sends back another .XML. We therefore had to implement a serialisation/deserialisation method between .XML and Kotlin, to send and interpret requests accordingly.
Secondly, the Google Maps API [4] was necessary to access accurate current location of the user as well as for internal functionalities such as converting addresses into latitude and longitude to be able to make the trip request. This API is available in Kotlin.

However, several functionalities of these two companies were not available as API and could not directly be implemented in the app: the live turn-by-turn navigation with audio guidance for Google Maps and the buying of tickets for SBB. Therefore, it was chosen to launch these apps on the correct window and to develop a notification system that would make navigation through the different application possible and fluid.

Considering these design choices, we implemented the following solutions. Most of the files we will discuss can be found in this directory: *Helpie2/app/src/main/java/com/example/helpie*.

A. **Global Theme**: The foundational theme framework is mostly done in *HelpieViewModel*, *HelpieScreen.kt* and *UiState.kt*. These files manage the ViewModel class, the foundational screen framework and navigation, and ViewModel variables, respectively. Each functionality relies mainly on these core UI files (see fig. 1). Furthermore, the app's core structure is maintained in *MainActivity.kt*. All UI (User Interface) elements are implemented using Kotlin Jetpack Compose objects such as buttons, boxes, texts, text fields, and others.

B. **Start and Destination Screen**: We can locate the start screen at *ui/StartScreen.kt*, which adheres to the global theme and features a prominent start button. Clicking on it triggers navigation to the destination page. Here, the UI design is implemented in *ui/DestinationScreen.kt*, while destination settings are managed in *ui/SettingsScreen.kt*, providing users with text fields to input new registered destinations. To ensure long-term usability, we store the UI state of destinations as JSON files.

C. **Journey Summary**: When a destination is entered, a resquest is sent to the the SBB API OJP trip request [5]. The resulting trip is stored and used for the summary. This is done in the *tripPlanificator* folder, utilizing the *tripPlanificator/OjpSdk* class and the *tripPlanificator/tripHandling.kt* file. Then, in *ui/SummaryScreen.kt*, we display the necessary information for a comprehensive summary and save trip information as step information class.

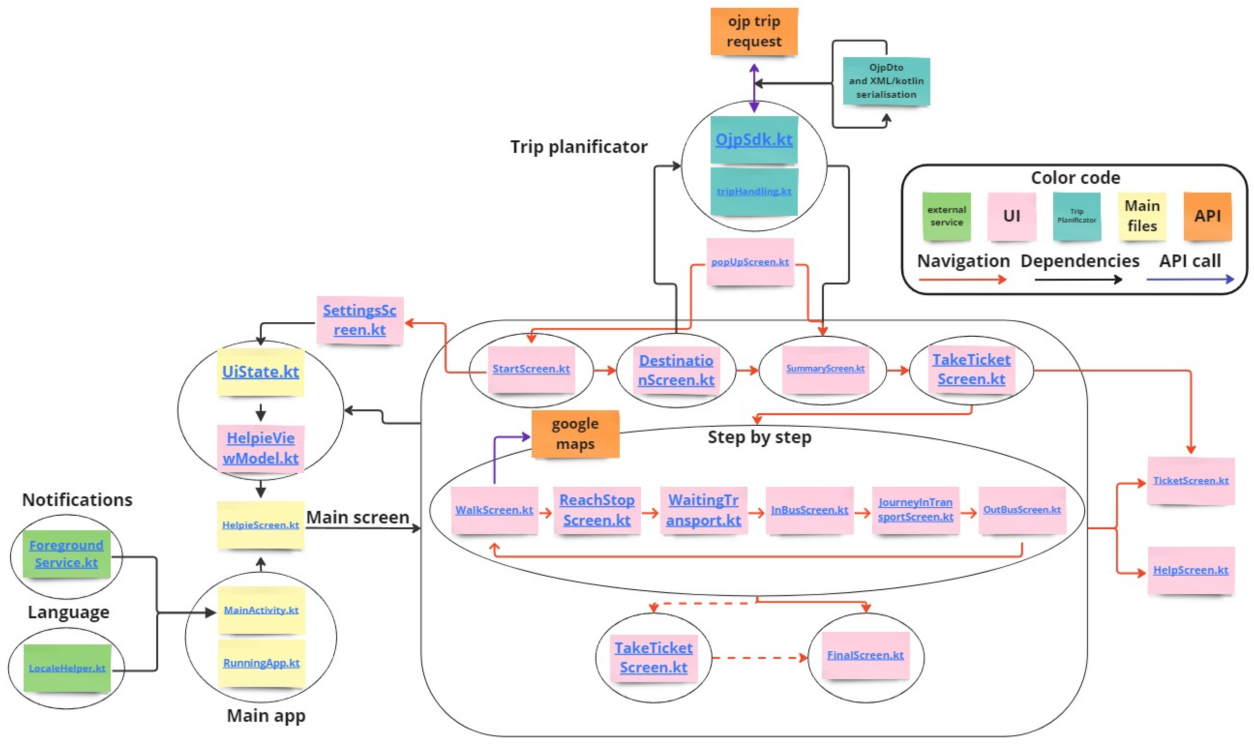


Figure 1: Repository graph, to understand the code structure

D. **Take a ticket**: To handle the two cases for the ticket, we use a boolean that can be modified in *ui/SettingsScreen.kt* via checkboxes. All UI elements are managed in *ui/TakeTicketScreen.kt*. In the case of *EasyRide*, we send a URL link to EasyRide, which transfers the corresponding *CFF* app feature. The file *ui/TicketScreen.kt* takes care of the page when a ticket is choosen and follows the same logic for taking the ticket. This screen is used at the start to take the ticket and at the end to desactivate Easyride if needed.

E. **Step by step instruction**: This features having many different pages to handle each travel steps, it required the following UI files:

- To handle stop verification:
  - *ui/InBusScren.kt*
  - *ui/OutBusScreen.kt*
  - *ui/ReachStopScreen.kt*
- Waiting screen before and while a trip:
  - *ui/JourneyInTransportScreen.kt*
  - *ui/WaitingTransport.kt*
  - *ui/WalkScreen.kt*

Each of these files uses information from the SBB API gathered previously in Journey Summary.

F. **Appropriate notifications and warning**: Both the window display and notification functionalities are designed within the *ForegroundService* class and are implemented inside *Main-*

Activity.kt, along with permission requests. We use the WindowManager and ForegroundService package for these purposes. The pop-up window, when a trip is interrupted midway, is handled in *ui/PopUpStop.kt*.

G. **Help button**: The help button is always shown by putting it in the global framework in *HelpieScreen.kt* and its screen UI is handled in *ui/HelpScreen.kt* as well as the change of app to call a number.

H. **Two interfaces**: We implemented a Settings screen using a button on the Start page. This screen is designed in *ui/SettingsScreen.kt* as mentioned earlier.

I. **Progress bar:** The progress bar is implemented in *HelpieScreen.kt* and consists of a Kotlin Jetpack compose drawable which adapts to changes in the UI state *current_step* using the *compose.runtime* function *LaunchedEffect*.

5 Conclusion

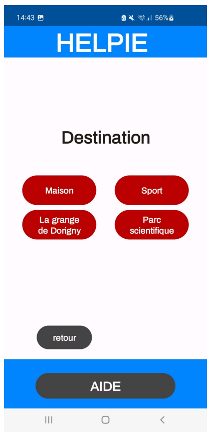
In the end, we were able to develop a working android app that is more intuitive, less overwhelming and more adapted to Charlie's needs. The app design and working was first pre-validated through a figma prototype. After development, the design and UI intuitiveness was confirmed by letting the challenger navigate through the app with little explanation and with a lot of fluidity. App changes between Helpie and SBB/Google Maps [3, 4] were easy and information balance was evaluated as very good: he did not feel overwhelmed nor thought about missing interactions.

Finally, the prototype was then validated through a real-life test of a travel that included walking, taking the metro and the bus. Charlie was able to follow the instructions and was very diligent about using the app. Overall, the app helped him orienting himself during the travel. However, as the low flexibility of the trip request - due to the API constraints - did not allow for taking into account delays, some confusion arose when the app was not perfectly in synchronisation with real time events. As accessing the data base of the SBB is not possible, this could be countered by having an additional interaction that could either take an earlier trip (when taking a late transport that was not considered by the app), or inform the app of the delay. Google maps was also not very efficient on complicated stations like Lausanne Flon, and other solutions could be investigated for these special cases.

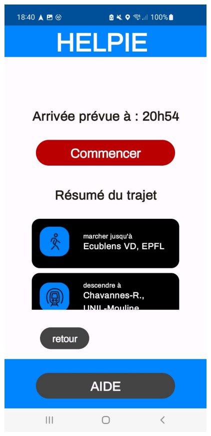
Overall, the app seems to fit the needs of Charlie, and we hope that, with the last adjustements, he will be able to travel more easily and more independently in the future.

Beyond the primary goal of this project to make public transports more accessible for Charlie, this app could be expanded into future projects to fit other profiles such as other people on the autistic spectrum, people with Down Syndrom or even for older people who feel lost on the original SBB app. As all these users can have some intersecting but also very different needs, the app was conceived to be modularised and personalised. Additional features could thus be inserted easily and allow more people to benefit from this app and be able to enjoy mobility as well as all its benefits.

6 Appendix



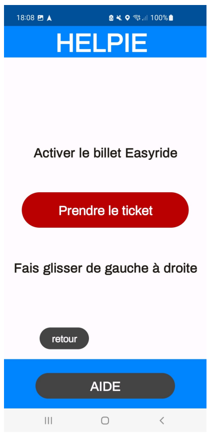
(a)


HELPIE
Commencer
marcher jusqu'à Ecublens VD, EPFL
descendre à Chavannes-R., UNIL-Mouline
retour
AIDE

Arrivée prévue à : 20h54

Résumé du trajet

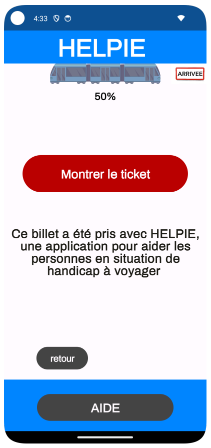
(b)



Activer le billet Easyride

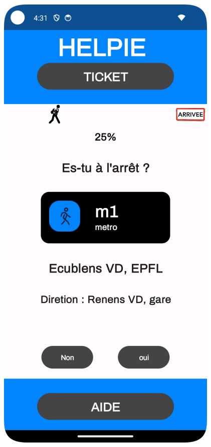
Fais glisser de gauche à droite

(c)



Ce billet a été pris avec HELPIE, une application pour aider les personnes en situation de handicap à voyager

(d)

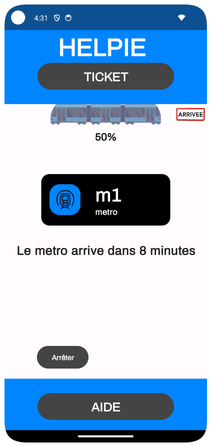


Es-tu à l'arrêt ?

Ecublens VD, EPFL

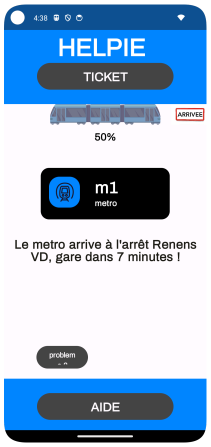
Diretion : Renens VD, gare

(e)



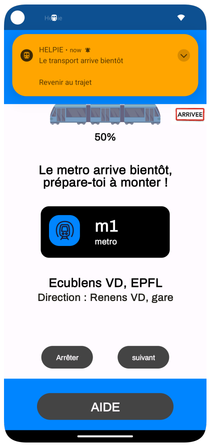
Le metro arrive dans 8 minutes

(f)



Le metro arrive à l'arrêt Renens VD, gare dans 7 minutes !

(g)



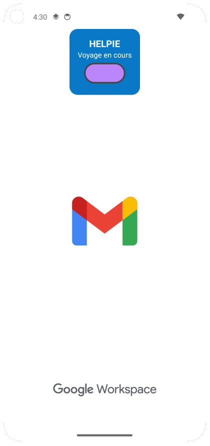
Le transport arrive bientôt

Le metro arrive bientôt, prépare-toi à monter !

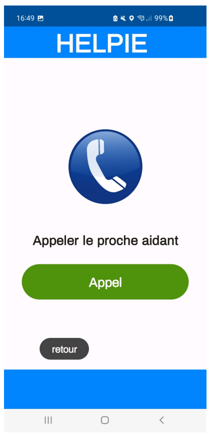
Ecublens VD, EPFL

Direction : Renens VD, gare

(h)

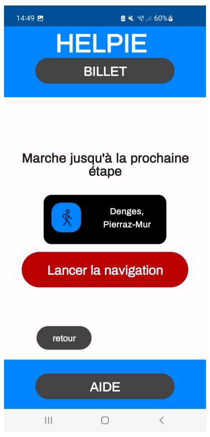


(i)



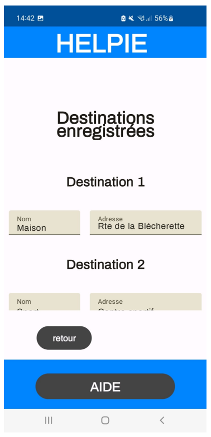
Appeler le proche aidant

(j)



Marche jusqu'à la prochaine étape

(k)



Destinations enregistrées

Destination 1

Nom: Maison — Adresse: Rte de la Blécherette

Destination 2

(l)

Figure 2: Helpie Screens

References

[1] *Top iPhone Travel Apps Ranking in Switzerland [3 June]*. Similarweb. URL: https://www.similarweb.com/top-apps/apple/switzerland/travel/ (visited on 06/07/2024).

[2] Jade Therras et al. *ATC-Helpie-2024*. Version 1.0.0. June 2024.

[3] *Open Journey Planner (OJP) — Plateforme open data pour la mobilité en Suisse*. URL: https://opentransportdata.swiss/fr/cookbook/open-journey-planner-ojp/ (visited on 06/07/2024).

[4] *Google Maps Platform Documentation — Maps SDK for Android*. Google for Developers. URL: https://developers.google.com/maps/documentation/android-sdk (visited on 06/07/2024).

[5] *OJPTripRequest — Open data platform mobility Switzerland*. URL: https://opentransportdata.swiss/en/cookbook/ojptriprequest/ (visited on 06/07/2024).