English project

English project report

Dreyer Mathieu, Jacque Antoine, Millardet Quentin

Academic Year 2019–2020

User guide and technological concepts

Academic supervisor: Fournier Karen

Contents

1 Introduction

This project was made in order to help non-English speakers to learn by playing. The aim is to ask simple questions to them and wait for a response. Depending on the question's difficulty, this can be more hard to find the correct answer. We imagine our application to be adaptive and versatile. In that way, questions can be added manually by a system administrator. As our project is designed like a little game, it can be proposed to children. The main design of the application is similar to actual mobile applications in order to have good ergonomics.

2 User Guide

2.1 Final user guide

This user's guide is intended for people wishing to learn English, and for teachers of English. The website is responsive, so it's accessible by phone, tablet and computer. The website can be accessed at this address: Pokelearning[5].
the user manual will be structured as follows :

- first part on the home page,
- for a second on how to play,
- and a last one on the administrator part

2.1.1 Homepage

On mobile phone Only

So when a user goes to the site, he comes to this home page. On mobile phone only, when arriving at the home page, it is possible to add the page to the home screen to return more easily to it.

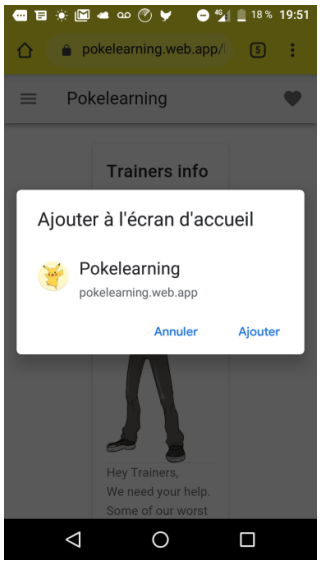

Figure 2.1: Add website on home screen

The user just has to choose whether or not to add the page to his home screen.
It's the only feature that's only on the phone.

On all devices

After that, we have a person who explains how it works.
In the different Pokemon games, a fairly old teacher explains the game to us.
We've chosen to have a trainer here to explain how to use the application.
So on the phone as well as on the computer we can see this character, with a little text on how the game works.

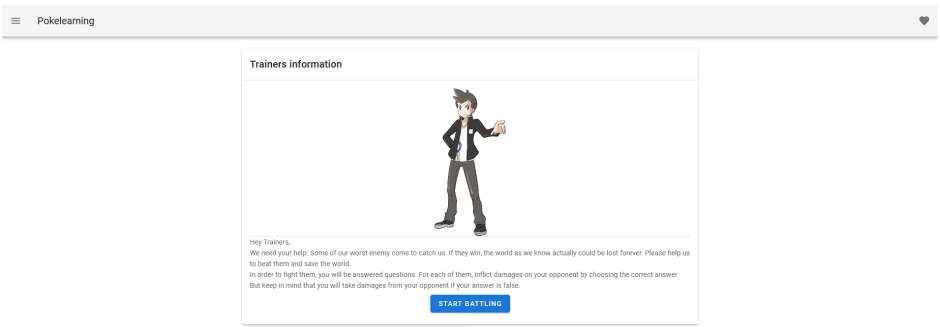

Figure 2.2: Homepage of the website on computer

In short, the trainer tells us here that we need help to beat our enemies and that to do so, we will have to answer various questions. With a good answer we take away health points from the enemy, while a bad answer takes away health points. After having taken knowledge of the functioning, and the principle of the game, it is necessary to click on the button "start battle".

2.1.2 In game controls

Once the user has clicked on "start battle", he is redirected to the battle page.
The design of the battles is similar to Pokemon.
On the top left are the opponent's hit points.
On the top right, an avatar representing the opponent is made.
In the middle right, the user's health points.
At the bottom right, a multiple choice option is available for the user. Among these choices there are

- attack : fight the trainer,
- objects : give heal or skip a question,
- help : explains how to play,
- flee : leave the game

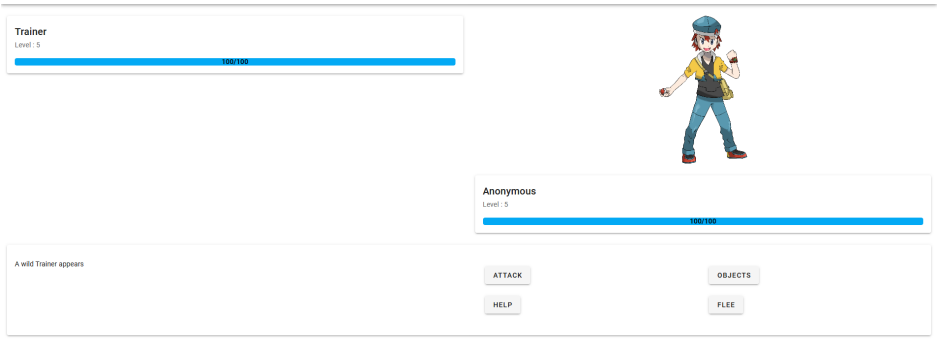

Figure 2.3: Battle on computer

When choosing a battle, the trainer disappears to make way for the question. When the question appears, it appears as a sentence with a missing word. There are 4 possible answers, a good answer takes away 30 HP from the opposing trainer, a bad one takes away 10. There is also a button to go back to the previous menu to use items or to quit the game.

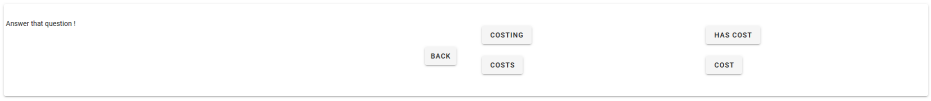

Figure 2.4: Response options

Once you have won or lost, the user can choose to restart a game, or stop there

2.1.3 Administration page

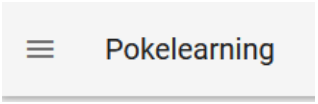

Figure 2.5: Bring up the menu

To be able to manage the administration part, you have to click on the button on the top left, then go to the administration category.
The administrator must then log in with his account name and password. (By default user name : admin, password : admin)

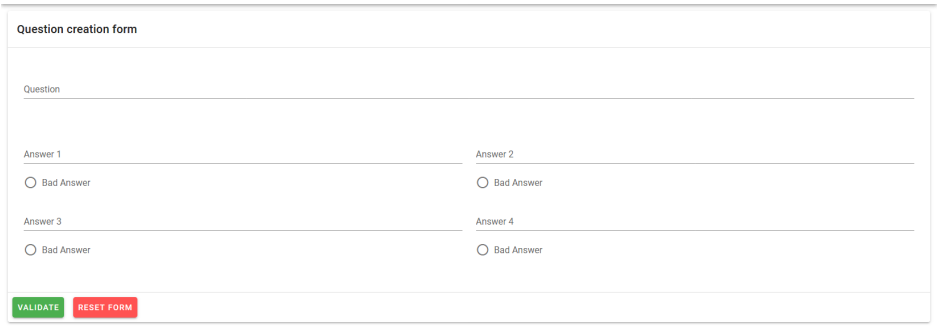

Figure 2.6: Administrator panel

Once the administrator has logged in he can add a question. To do so, all he has to do is fill in his question or phrase with the missing word in the question field, replacing the missing word with "_" (for a good harmonization, it is recommended to put 6). Then in the answer part, you have to enter 4 choices, including a good answer and 3 wrong answers. Finally, you have to define the right answer by pressing the button below the answer. At the end of the question entry, there should be 3 "bad answer" and 1 "good answer".

3 Deployment Guide

3.1 Requirements

In order to facilitate the deployment of this application, we build it entirely as a front application. For this application, and in order to deploy it, you will need a firebase account (the free account will be enough).

In order to start the deployment process, you will need a computer meeting those requirements :

- Have an Internet connection
- Have a git client in order to clone the sources or an archive containing all sources
- Have the possibility to install and build npm dependencies (npm client)

If you don't want to use the firebase services, you will need more requirements :

- An accessible web server
- A valid HTTPS implementation on this server
- A database and an API that permit data query and update (only if you don't use firebase database)

3.2 Installation guide

The first thing you have to do is clone the sources. In order to do this, open a terminal in the development directory and type :

```
1 git clone "git@gitlab.telecomnancy.univ-lorraine.fr:pokelearning.git"
```

But if you have the archive containing the sources, you can simply extract the sources in your working directory.

Now, you need to install the dependencies of the project. Use the following npm command :

```
1 npm install
```

Depending on your connection, this can take some minutes. This command will install every external dependency in the "node_modules" sub-directory.

Because all of the code used in our project is heavy, and in order to compress what the server will send to the client, we use a gathering module : Webpack. To run the gathering and generate the effective frontend code, you have to run the following command :

```
npm run build
```

3.3 Deploy on firebase

In order to deploy the web application on your firebase hosting, you have to create a firebase account (or use an existing one) following this link (Note : if you have a google account you can use it). Now, go to the console view and create a new project by clicking on the screen. After filling the different pages of the form, you will be redirected on the project overview. Now go on the Hosting page (by clicking in the left panel on the corresponding menu). Click on the "start" button and follow the 3 next instruction given by the platform. By executing the last command, the application will be send on the hosting page and put online. The url of your application will be :

```
"https://" + name of your project + ".web.app"
```

This url will be printed in return of your last command additionally.

The next step is to add new questions to the real-time database. Go to the firebase console in the database menu. Create a new realtime database by following the instructions given by the plateform. Add a new node in the tree named "Questions". you don't need to manually add questions, a dedicated administration interface will help you.

3.4 Deploy on a server

The first thing you have to do is modify the procedure to get the questions in the local database. In order to do that, you need to modify the file

`\src\plugins\IDBPlugin\IdbPlugin.js"`

In this file, you have to replace this code by your own that query your database and format the question as it's expected :

```
firebase.app().database().ref('Questions').on('value', function (num_question)
    {
            let questions = num_question.val();
            for (let key in questions) {
                let question = questions[key];
                if (question instanceof Object) {
                    question.id = key;
                    indexedDb.put("questions", question)
                }
            }
```

```
10          });
```

Listing 3.1: lines of code to replace in order to set up a personal database system

After modify the document, you need to rebuild the project files using this command :

```
1 npm run build
```

To finish you need to copy the "dist" folder on your server. Then create a new virtual host (or similar) that points to the "index.html" file located in the "dist" directory. The way to do this depends on your server, so i can not explain procedure in details.

4 Technical implementation

4.1 VueJs

Dans le cadre de ce projet, nous avons utilisé VueJS : un framework JavaScript open-source créé pour construire des interfaces utilisateur et des applications web monopages. Pour réduire au maximum les échanges avec le serveur, Il est possible de créer une application complète monopage *SPA*. Une SPA est chargée entièrement en une fois et n'est jamais rafraîchie. Toutes les modifications de l'interface sont faites du côté du client à l'aide de scripts. Ainsi une telle application est chargée une fois puis peut être utilisée même hors connexion. L'utilisation d'une SPA simplifie l'interaction avec un serveur et nous a permis de charger toute l'application en une seule requête web. Ce système pourra donner l'impression d'une navigation dans l'application à l'utilisateur tout en étant hors ligne.

Le principe de Vue js est de permettre de créer des composants graphiques réactifs et encapsulables les uns dans les autres. La réactivité est caractérisée par le fait que l'affichage est actualisé lors d'un changement de valeur du modèle. De plus, du fait que les composants graphiques sont indépendants les uns des autres, lors d'une modification du modèle, seuls les composants impactés sont actualisés.

4.1.1 Vuetify

Vuetify est un plugin de VueJS permettant d'ajouter des composants graphiques complets, ergonomiques et paramétrables. Il permet en plus de gérer des formulaires avec règles de vérifications, de structurer une interface de façon claire et de gérer tous les aspects de style rapidement.

Pour le développement de notre application, nous avons principalement utilisé les structures Grid et Card permettant de garder la main de manière responsive sur chaque partie de l'écran.

Le style par défaut de ces composants est basé sur les visuels de Material Designs utilisés pour les applications Android. Ceci permet de rendre l'application agréable à utiliser mais permet aussi de garder les réflexes des utilisateurs habitués à ce type de fonctionnement.

4.1.2 Vue-router

Vue-router est un second plugin de VueJS couramment utilisé. Il permet principalement de rendre l'application navigable en changeant dynamiquement l'affichage des différents composants graphiques. Cet outil permet de donner à l'utilisateur une impression de navigation classique alors qu'il ne fait que modifier l'affichage au moyen de scripts. De plus comme toute la navi-

gation se fait de manière locale, aucune attente n'est visible car aucune requête vers un serveur n'est effectué pour recréer la page. Il est possible de définir les différentes routes possibles dans la configuration de ce plugin ainsi que des redirections et des sous-routes comme nous pouvons le voir dans le code ci-dessous. 4.1

```
export default new Router({
    mode: 'history',
    routes: [
        {
            path: '/',
            redirect: {name: 'home'}
        },
        {
            path: "/home",
            component: Home,
            name: "home"
        },
        {
            path: '/battle',
            component : Battle,
            name: "battle"
        },
        {
            path: '/newQuestion',
            component: NewQuestion,
            name: "admin"
        }
    ]
})
```

Listing 4.1: configuration du routeur VueJs

4.2 Progressive web app et service worker

Afin de rendre notre application utilisable rapidement sur les terminaux mobiles, nous avons choisi d'intégrer le système de progressive web app à celle-ci. Cette option permet de rendre notre application installable sur le terminal que l'on utilise.

Une pwa est définie comme étant la frontière entre une application native et une application web. D'après la documentation Google developper [3] une pwa offre une expérience d'application native installable tout en étant construite pour du web. Elle a pour but d'être rapide et fiable. L'article paru dans "medium" en mai 2018 [2] décrit les fondements d'une pwa. Afin de combler l'écart entre une application native et une application web, une pwa embarque un fichier Manifest.json (ou WebApp Manifest). Celui-ci définit les métadonnées manquantes à l'application web pour apparaître comme une application native telles que l'icône principale ou le nom court de l'application. D'après ce même article [10], le WebApp Manifest est en cours d'implémentation sur tous les navigateurs majoritairement utilisés (d'après la source [4] il est complètement implémenté sur Firefox). Le Manifest et ses spécifications sont définies par la W3C.

Lors de la première visite sur l'application web, la pwa va proposer de l'ajouter à l'écran d'accueil (sur les terminaux mobiles) ou bien une icône d'installation sera affichée dans la barre de navigation (navigateurs PC). Une fois installée elle apparaît alors dans la liste des applications disponibles sur le terminal mobile.

Pinterest, Twitter et L'Equipe ont tous les 3 mis en place des pwa pour leurs services par exemple.

En plus de pouvoir télécharger l'application, nous avons intégré un système de service worker permettant de gérer de manière personnalisée le cache et les appels à celui-ci. Il s'agit d'un script qui s'exécute en parallèle du chargement de la page web et qui va permettre d'intercepter les requêtes vers l'extérieur pour les traiter. Il se place alors en proxy de l'application web. Il permet notamment de placer les pages chargées en cache et de rediriger les requêtes externes vers ce cache permettant ainsi de garder une version offline du site visité. L'utilisateur a donc l'impression d'une navigation en ligne tout en étant hors ligne.

Le couplage ce cette architecture avec le framework de développement monopage Vuejs permet de charger la totalité de l'application en cache dès le premier accès au site.

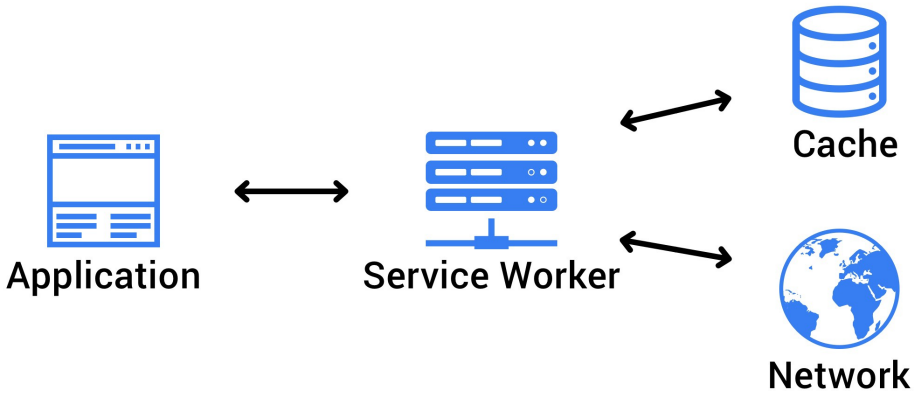

Figure 4.1: Position du ServiceWorker dans une application web, crédits [6]

4.3 IndexedDb

Il s'agit d'une API permettant de stocker des information à la manière d'un Système de gestion de base de données(SGBD). La définition donnée par Mozilla [1] indique que ce système de gestion de données permet de stocker des objets javascripts. IndexedDb n'est pas basé sur des tables contrairement aux SGBD classiques mais sur des ObjectStores. Chaque ObjectStore permet de stocker des objets partageant une même clé. La clé en question permettra de retrouver les objets par la suite. Il est possible de définir des index supplémentaires afin de rechercher des objets sur d'autres propriétés. Ce système de stockage est donc optimisé pour la recherche et l'ajout d'objets structurés. Toutes les transactions sont faites de manière asynchrones de façon à ne pas bloquer l'exécution d'un script lors d'une recherche ou d'un ajout de grand nombre de données. Google chrome et firefox proposent de plus les outils de développement permettant de visualiser un ObjectStore ainsi que tout son contenu 4.2.

On peut noter que les navigateurs actuels sont tous compatibles avec cette API. IndexedDb est aussi fonctionnel au travers des navigateurs embarqués par les webView Android. Cette solution, en plus d'être adaptée à la problématique est efficace. L'utilisation de cette API est limitée suivant la quantité d'information à stocker mais largement acceptable pour notre problématique de

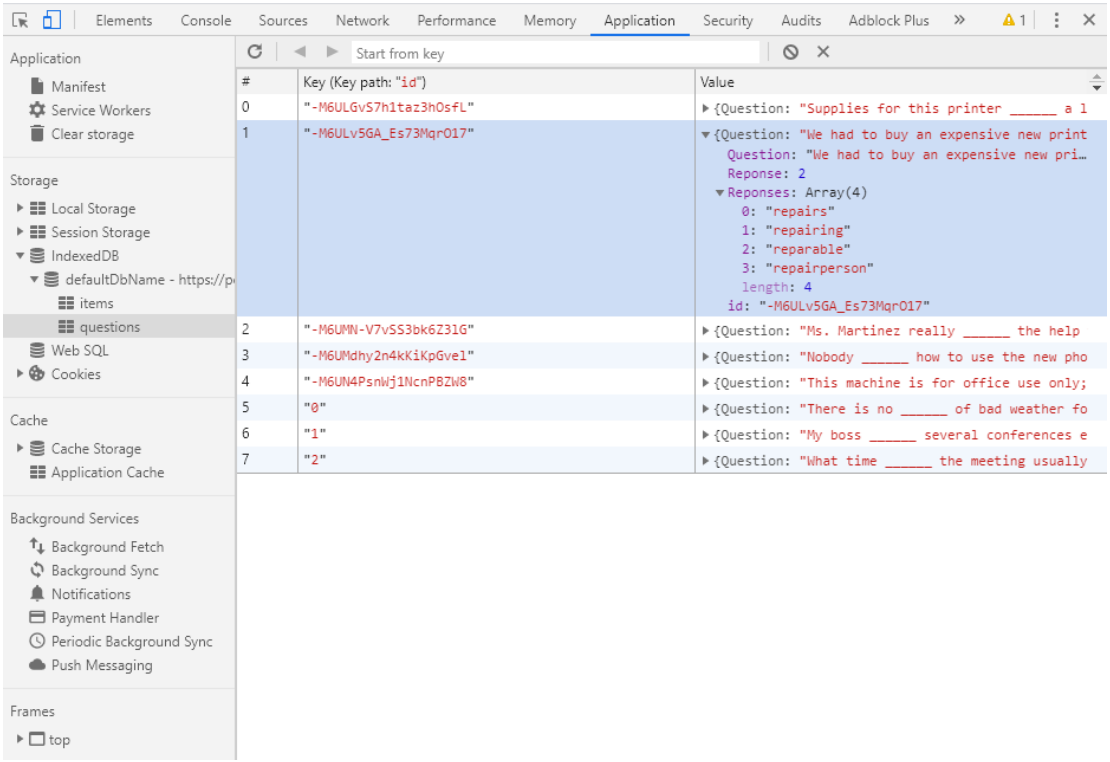

Figure 4.2: Vue des outils de développement google chrome pour IndexedDb

rendre une application utilisable hors ligne.

4.4 Firebase

Firebase est un système d'hébergement et de services de Google. Ces services sont gratuits pour des besoins léger mais peuvent devenir payants pour des besoins plus importants. Dans le cadre de notre projet, nous avons utilisé deux outils de cette solution : Firebase Hosting et Firebase Realtime Database. Il en existe bien d'autres, la liste complète est disponible ici

4.4.1 Hosting

Pour les besoins de notre application, nous avons pu nous contenter de la version gratuite des service de firebase hosting. celle ci permet notamment de fournir

- Un hébergement web
- Un accès en HTTPS avec un certificat valide.
- Un nom de domaine

Du fait de l'utilisation d'une progressive web app, le protocole HTTPS était nécessaire.

De plus, firebase est fourni avec un ensemble d'outils permettant le déploiement facilité, une fois la phase de configuration terminée.

L'ensemble de ces outils nous permettent de déployer le code source de l'application en une seule commande.

4.4.2 Real-time database

Firebase Realtime Database est une solution d'hébergement de données au format JSON. Ces données sont accessibles par tous à tout moment selon des règles définies lors du déploiement de l'application. c'est le principe d'une base de données temps-réel. Chacun des clients va utiliser les mêmes données. Ce système permet de changer en direct les données pour tous. De plus il permet d'avertir les multiples clients qund une modification est apportée (d'où le terme real-time)

Google fourni tous les outils nécessaires pour assurer la liaison avec cet outil framework js.

5 Conclusion

The main goal of the project is to build an application to help the learning of the English language. We have therefore chosen to do it in the form of a question/answer, or a sentence with a missing word. We also wanted to make it fun, and attractive.
To do so, we chose to hijack a video game that the whole group knows : the Pokemon game.

This project does not represent the finality but an example of what can be done. Because, in one hand, you can manage questions on your own to create more or less difficulties. And on the other hand, it can be simply deployed on an other instance to have multiples sets of questions.
We based our work on the look and feel of the game. An for the administrator point of view, we build a quick and easy form to add a question.

Several improvements are possible on the application, which would make it more efficient. With more means and time, we could have added an account system, to allow a follow-up of the users, to see the progress. We could also have defined a level of difficulty for users, so that they could choose more complex questions.

This project both broadened our knowledge base and strengthened it for those who had already used a similar tool.

Bibliography

[1] alattalatta (Jongryul Yang). Indexeddb api. https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API. 12

[2] Guillaume André. Progressive web app (pwa) — support et compatibilité des web app manifest, mai 2018. https://frama.link/RwdShfK-. 11

[3] Google. Your first progressive web app. https://frama.link/VofZu3U_. 11

[4] Mozilla. firefox plateform status. https://platform-status.mozilla.org/#app-manifest. 11

[5] us. pokelearning. https://pokelearning.firebaseapp.com/home. 3

[6] Alexander Zlatkov. How javascript works: Service workers, février 2018. https://blog.sessionstack.com/how-javascript-works-service-workers-their-life-cycle-and-use-cas
12, 17

List of Figures