

# Rapport du projet Watizit

Nicolas Lauret, Andriniaina Randrianasolo

17 mai 2020

The logo for 'WATIZIT' features the word in a bold, sans-serif font. The letters 'W', 'A', 'T', 'I', and 'Z' are a vibrant blue, while 'I' and 'T' are a bright pink. The text is set against a dark blue background with several overlapping, semi-transparent circles in various shades of blue and purple, creating a modern, abstract feel.

# WATIZIT

Try to find things from their shape

## Résumé

Watizit est un jeu mobile disponible sur Android dont le but est de retrouver le nom d'un animal ou d'un objet à partir de sa forme. Dans ce rapport, nous allons vous présenter en détails notre projet effectué durant notre formation en L3 Informatique à l'Université de la Réunion.

*Nous tenons à remercier toutes les personnes qui ont pris le temps de tester notre application tout au long de l'avancement de notre projet et qui nous ont fait part de leur retour.*

*We would like to thank all the people who took the time to test our application throughout the progress of our project and who shared their feedback with us.*

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Description générale</b>	<b>3</b>
<b>3</b>	<b>Organisation de travail</b>	<b>4</b>
<b>4</b>	<b>Design</b>	<b>4</b>
<b>5</b>	<b>Animations</b>	<b>5</b>
<b>6</b>	<b>Interactions et données</b>	<b>7</b>
<b>7</b>	<b>Gameplay</b>	<b>9</b>
<b>8</b>	<b>Musique</b>	<b>10</b>
<b>9</b>	<b>Accessibilité</b>	<b>10</b>
<b>10</b>	<b>Autres fonctionnalités</b>	<b>11</b>
<b>11</b>	<b>Code source</b>	<b>11</b>
<b>12</b>	<b>Conclusion</b>	<b>12</b>

# 1 Introduction

Grâce aux technologies et aux outils accessibles de nos jours, il est devenu de plus en plus simple de se lancer dans la création d'applications et de jeux. Mais étant donné que ce nouveau monde est accessible au grand public, il devient de plus en plus difficile de se faire une place parmi les nombreuses applications publiées chaque jour.

Ayant des expériences antérieures dans la création de jeux vidéo, nous nous sommes donnés comme challenge de créer un jeu avec une idée novatrice qui pourrait trouver sa place au sein du marché. C'est ainsi qu'est né Watizit, une application qui reprend les fondements de base d'un jeu en y ajoutant notre idée : celle de retrouver le nom d'un animal ou d'un objet grâce à sa forme.

Au cours de ce rapport, nous aborderons les points importants de conception qui nous ont permis de mener à bien notre projet.

## À qui s'adresse cette application ?

Notre jeu a été créé dans l'optique d'être simple et rapide à prendre en main, sans demander trop d'effort de concentration, s'adressant ainsi à tout public à partir de 10 ans qui souhaite se divertir, dans les transports en commun par exemple, sur un jeu addictif où les niveaux s'enchaînent.

Il est disponible sur Android Jelly Bean (API 16) et plus, en français, en anglais et en espagnol, élargissant ainsi la potentielle audience, et offrant aux joueurs la capacité de tester leur vocabulaire dans d'autres langues.

# 2 Description générale

L'application Watizit a été réalisée à l'aide d'Android Studio, dont nos principales sources d'informations ont été la documentation Android [1] ainsi que les transparents de cours mis à notre disposition.

Réalisée dans un style moderne et disponible en version portrait et paysage, elle est composée d'un menu principal, un menu des options permettant de changer la langue de l'application ainsi que le volume de la musique de fond, un menu listant les 32 niveaux que comporte notre application et un menu permettant de jouer au niveau, en plus de deux fenêtres *popups* permettant respectivement d'acheter des indices et d'afficher l'écran de victoire.

Les niveaux peuvent se trouver dans 3 états différents : *bloqué*, *débloqué* ou *terminé*. Pour débloquer un niveau, il faut terminer le niveau précédent. Afin d'aider les joueurs qui auraient des difficultés à terminer un niveau, 3 éclairs (monnaie en jeu) sont offerts au joueur lors du premier lancement de l'application, qu'il pourra dépenser en achetant des indices.

Toutes les ressources que nous avons utilisées qui ne sont pas de notre création (images et musique) sont délivrées sous licence CC 3.0. En effet, les images proviennent du site *Flaticon* [2] et la musique, réalisée par HorrorPen, est disponible sur le site *OpenGameArt* [3].

### 3 Organisation de travail

Afin d'être le plus productif possible, nous avons mis en place dès le début du projet des outils qui allaient nous permettre d'être plus efficace par la suite, à savoir :

- Git et Github pour la gestion du développement de Watizit
- Trello pour la répartition des tâches, les priorités de développement et les deadlines
- Discord pour communiquer et faire le point au cours de l'avancement du projet
- Figma pour le design/prototypage des écrans de l'application

En plus de cela, nous avons effectué une phase de recherche en amont afin d'observer les jeux de la même catégorie que le notre et vérifier qu'il n'a pas déjà été fait et éviter de devoir changer d'idée en cours de développement.

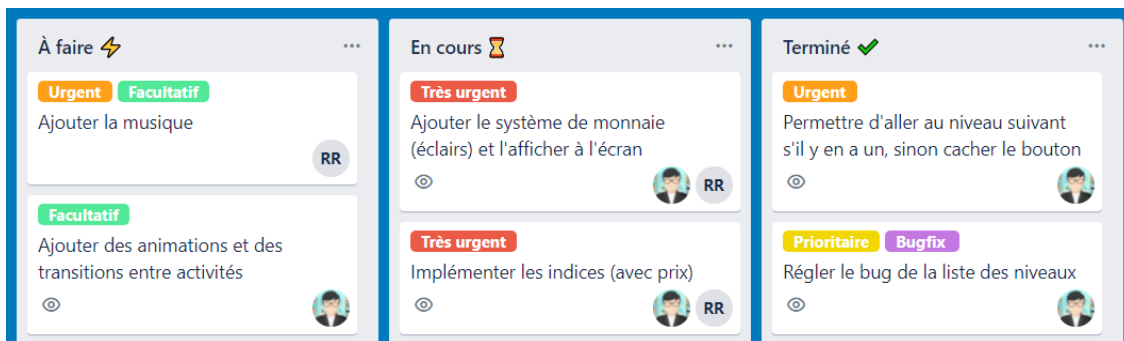


FIGURE 1 – Aperçu de notre répartition des tâches sur Trello

### 4 Design

Après avoir choisi l'idée de notre projet, nous avons commencé à prototyper les différents écrans de notre application sur Figma, outil collaboratif de design d'interfaces. Un premier design sommaire a été réalisé directement sur Android Studio (v1.0) afin d'avoir un rendu visuel de l'application. Au fil du temps l'application a changé de design (v1.1 puis v1.2) jusqu'à son apparence finale (v2.0).

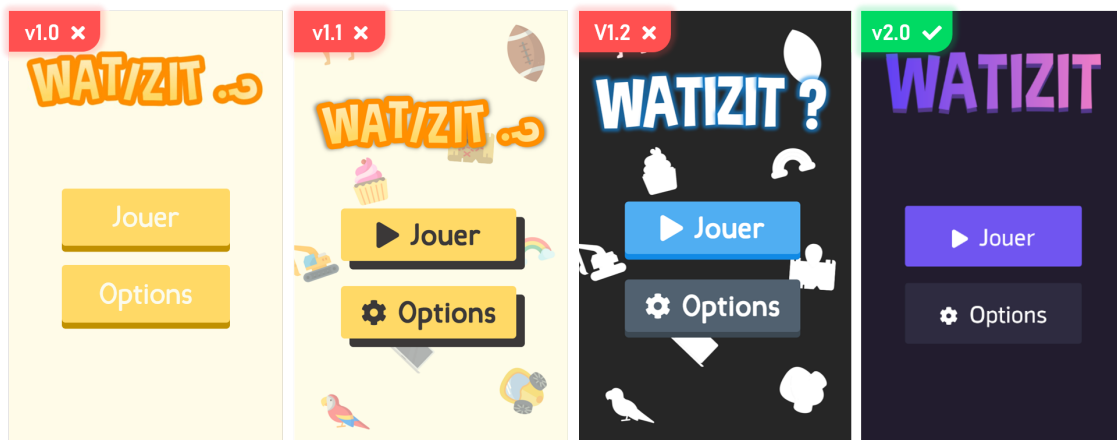


FIGURE 2 – Évolution des designs réalisés sur Figma

Nous nous sommes donc dirigés vers un design simple, moderne et flat afin d'attirer l'attention de l'utilisateur sur l'essentiel et d'éviter les informations inutiles pour ainsi réduire la charge cognitive de l'utilisateur.

Le choix des couleurs est expliqué plus en détails dans la section Accessibilité.

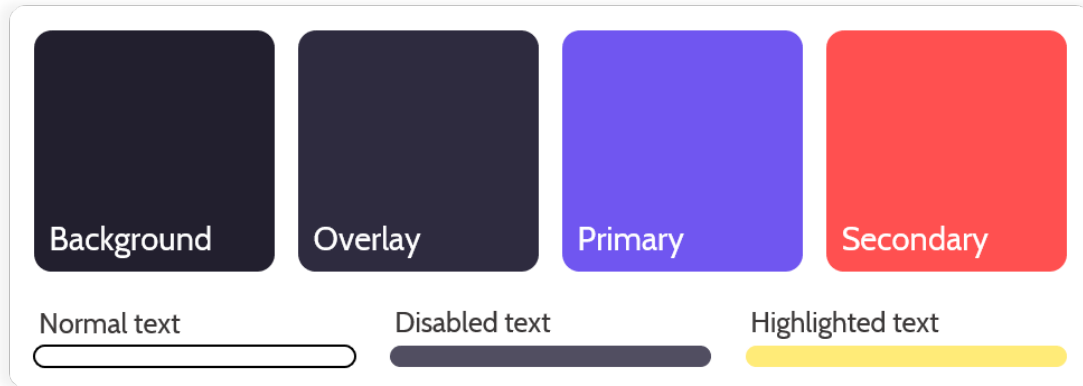


FIGURE 3 – Palette de couleurs de l'application

#### 🌙 Pourquoi avons-nous choisi un thème sombre ?

- pour réduire la fatigue oculaire dans les endroits sombres (39% des Français utilisent leur smartphone ou tablette au lit d'après une enquête de l'INSV en 2016 [4]) ;
- pour réduire la quantité de lumière bleue de l'écran, néfaste pour la vue et le sommeil ;
- pour optimiser le contraste de l'application, ce qui favorise l'accessibilité.

## 5 Animations

Afin de rendre l'application plus vivante et dynamique, nous avons ajouté des animations d'apparition sur certains boutons ainsi que certaines images de notre application. Pour ce faire, nous avons réalisé une animation de grossissement de la vue depuis une taille de 0% jusqu'à sa taille finale. Afin d'ajouter un effet "élastique", de rebondissement, pour donner plus d'impact et de réalisme à l'animation, nous y avons ajouté une fonction d'accélération de la forme :

$$f(t) = -(e^{-\frac{t}{a}} \times \cos(t \times w)) + 1$$

avec  $t$  le temps,  $a$  l'amplitude et  $w$  la fréquence.

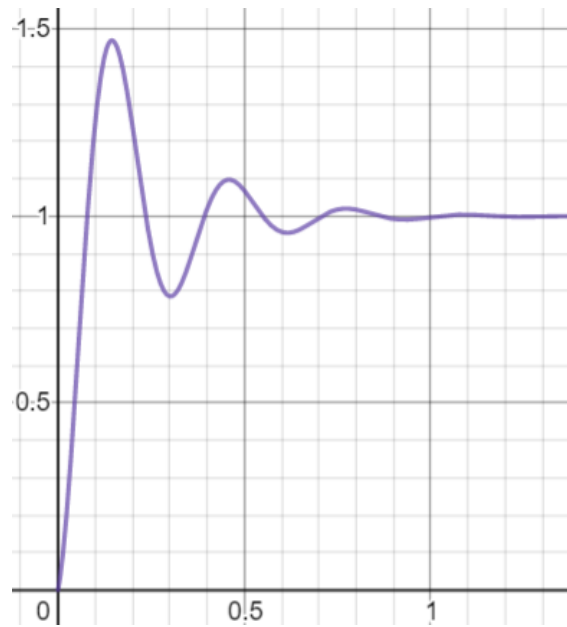


FIGURE 4 – Représentation graphique de la fonction d'accélération

#### Transitions entre menus !

En plus des animations de boutons et d'images, nous avons ajouté des effets de transition entre les différents menus afin d'avoir une navigation plus fluide et agréable visuellement.

Listing 1 – Exemple d'animation pour la transition d'un menu

```
<?xml version="1.0" encoding="utf-8"?>
<set
    xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:duration="@android:integer/config_shortAnimTime"
        android:fromXDelta="-100%p"
        android:toXDelta="0" />
    </set>
```

## 6 Interactions et données

Notre jeu comporte de nombreux menus et popups qui interagissent entre eux, souvent avec des échanges d'informations. Afin de mieux comprendre comment chacun d'entre eux communique avec les autres, nous avons réalisé un organigramme :

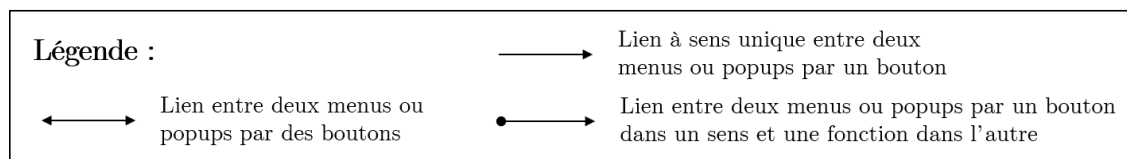
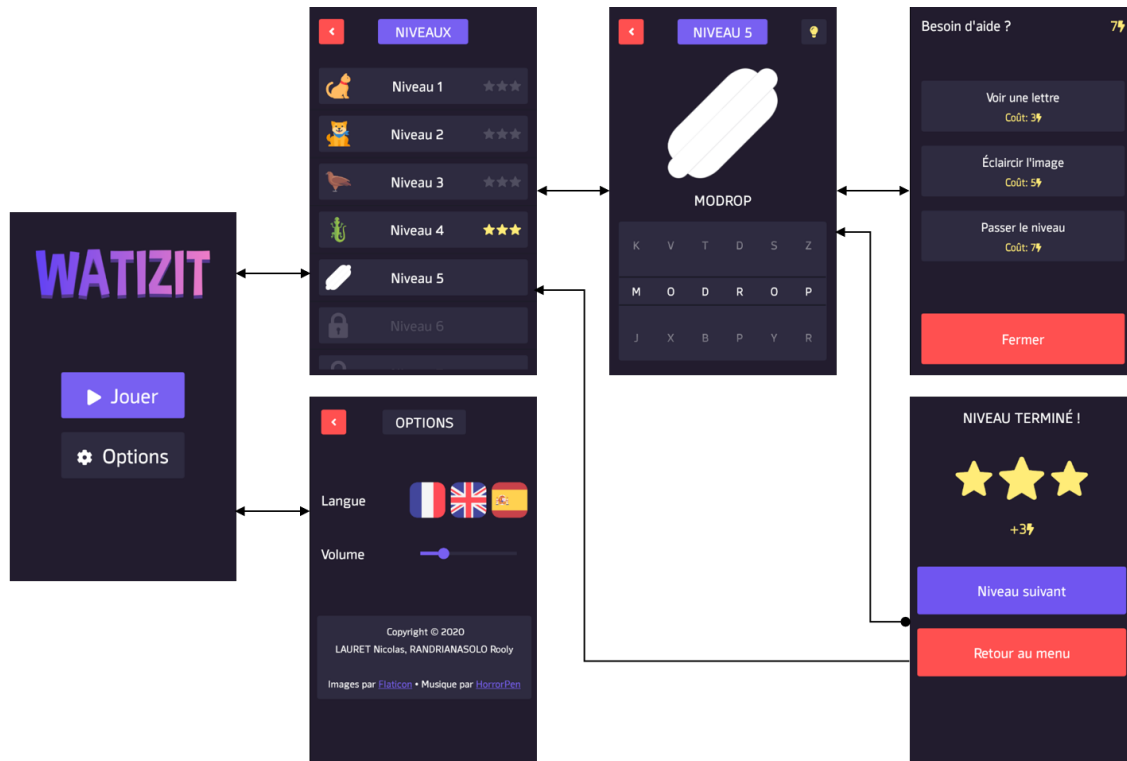


FIGURE 5 – Organigramme des interactions entre les écrans de l'application

Listing 2 – Interaction par fonction entre le menu du niveau et le popup de victoire

```
private void triggerWin() {
    ...
    // Create the win popup
    final WinPopup win_popup = new WinPopup(this, level);
    // And show it after a 1s delay
    // (to allow the player to have time to see the solution)
    new Handler().postDelayed(new Runnable() {
        @Override
        public void run() {
            win_popup.show();
        }
    }, 1000);
}
```

### ☰ Quelles données sont manipulées et stockées ?

Les petites données telles que la langue, le volume de la musique et la monnaie en jeu du joueur sont stockées dans les *SharedPreferences* de l'application, car elles ne nécessitent pas un large espace de stockage.

En revanche, les niveaux (qui seront de plus en plus nombreux au fil des mises à jour) ainsi que les données qui leur sont associés sont stockés dans une base de données.

Lorsque l'utilisateur clique sur un niveau de la liste des niveaux, ce dernier affiche le menu du niveau en lui communiquant le numéro du niveau à afficher. Afin de réaliser cela, nous envoyons un message dans la requête d'accès au menu comme suit :

Listing 3 – Envoi de l'information vers le menu du niveau

```
// When a level cell is clicked, check if it has an associated
// level and if the level is available
levelsList.setOnItemClickListener(new
    AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int
            position, long id) {
            Level level = (Level) parent.getAdapter().getItem(position);
            // If level isn't done and isn't locked, we can play it
            if ( !(level.isDone() || level.isLocked()) ) {
                // So go to the level menu activity and send the level ID
                // to the activity
                Intent intent = new Intent(LevelsListMenu.this,
                    LevelMenu.class);
                intent.putExtra("EXTRA_ID", level.getID());
                startActivity(intent);
            }
        }
    });
```

Listing 4 – Récupération de l'information par le menu du niveau

```
// Get level ID sent by the levels list activity
// Default ID = 0 if ID not found
int levelId = getIntent().getIntExtra("EXTRA_ID", 0);
// If ID < 1, it's not part of database so we have nothing to show
// and we finish the activity
if (levelId < 1) {
    finish();
    return;
}
```

Pour faire le lien entre un menu et un popup, pour appliquer un indice lors de l'achat par exemple, nous devons nous aider d'une méthode d'une *interface* de la classe du popup qui sera utilisée dans cette classe et qui sera écrasée dans le menu pour appliquer les changements. En voici un exemple ci-dessous :



Listing 5 – Interface et utilisation de la méthode dans le popup

```
hintButton1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        dismiss();
        // This will take the money from the player and update database
        level.buyHint(1);
        // This will apply the hint on the level menu
        listener.applyHint(1);
    }
});

public interface HintsListener {
    void applyHint(int hintNumber);
}
```

Listing 6 – Méthode écrasée dans le menu du niveau

```
@Override
public void applyHint(int hintNumber) {
    switch (hintNumber) {
        ...
        case 3: // Hint: Skip the level
            // Just call triggerWin() and the magic happens!
            triggerWin();
            break;
    }
}
```

## 7 Gameplay

Le jeu, dont le but est de trouver le nom d'un animal ou d'un objet à l'aide de sa forme, s'enchaîne sous forme de petits niveaux. Pour chaque lettre du mot à trouver, 4 lettres uniques sont choisies aléatoirement dans l'alphabet afin de proposer une roue de possibilités de 5 lettres au total pour chaque lettre du mot. À noter qu'une sécurité a été rajoutée afin d'éviter que lors du mélange des lettres, le mot n'apparaisse pas directement. Le but du joueur est de trouver la bonne lettre de chaque roue, afin de reconstituer le mot.

### ★ Un peu de challenge !

Deux éléments importants qui permettent de rendre un jeu attirant sont un objectif et une récompense :

- Un objectif caché de temps incite le joueur à trouver rapidement la solution au niveau, car sa récompense en sera impactée : plus vite il trouve le mot, meilleure sera la récompense. À la fin du niveau, le score du joueur est affiché sous forme d'étoiles en fonction de son temps et des indices qu'il a utilisés.
- La récompense quand à elle, est matérialisée sous forme d'éclairs, une monnaie en jeu permettant d'acheter des indices pour aider les joueurs en difficulté. Le joueur gagne autant d'éclairs qu'il a d'étoiles à la fin du niveau. Attention cependant ! Les indices ajoutent des malus de temps au score final.

Afin d'éviter un abus d'indices, nous avons inventé une formule qui applique un malus de temps au temps final pris par le joueur pour terminer le niveau :

$$\text{Soit } p(i) = \begin{cases} 2^{(i-1)} & \text{si l'indice a été utilisé} \\ 0 & \text{sinon} \end{cases} \quad \text{le poids d'un indice, on a :}$$

$$\text{tempsFinal} = \text{tempsDuNiveau} + 12000 \times \sum_{i=1}^3 p(i) \quad \text{en } ms \text{ (converti par la suite en } s)$$

Le coefficient 12000 a été choisi pour éviter qu'un joueur utilisant l'indice "Passer le niveau" n'obtienne le score maximal de 3 étoiles ; en effet, s'il utilise cet indice au début du niveau, on a  $\sum_{i=1}^3 p(i) = p(3) = 4$ , donc  $\text{tempsFinal} = 0 + 12000 \times 4 = 48000ms = 48s$ , or pour gagner 3 étoiles il faut finir le niveau en moins de 45 secondes.

## 8 Musique

Afin de rendre notre jeu plus vivant, nous avons décidé d'y ajouter une musique de fond. Pour ce faire, nous avons dû créer un *Service*, composant permettant de réaliser une opération de longue durée sans interaction avec l'utilisateur. Ce service qui s'exécute en fond sur l'entiereté de l'application contenait alors un *MediaPlayer*, composant permettant de lire un fichier audio. Avec ces deux éléments, nous avons une musique de fond.

Mais nous ne nous sommes pas arrêtés là, en effet il fallait laisser à l'utilisateur l'opportunité de changer le volume de la musique, et pour cela nous avons donc dû créer un lien entre le *MediaPlayer*, le *Service*, les *SharedPreferences* et le *SeekBar* (slider) du menu des options. Étant donné que le slider pour le pourcentage de volume de la musique peut aller de 0 à 100, mais que le volume du *MediaPlayer* ne va que de 0.0 à 1.0, nous avons dû utiliser une fonction logarithmique afin d'avoir une échelle de volume linéaire :

$$f(\text{volume}) = 1 - \frac{\log(\text{max}-\text{min}+1-\text{volume})}{\log(\text{max}-\text{min}+1)} \quad \text{avec } \text{max} = 100 \text{ et } \text{min} = 0 \text{ dans notre cas}$$

Une fois cela fait, afin de mettre la musique en pause lorsque l'utilisateur quitte l'application ou que celle-ci n'est pas *focused* (si l'utilisateur regarde les applications en cours d'exécution à l'aide du bouton *Home*), nous avons dû ajouter un *HomeWatcher* qui écoute les actions de retour à l'accueil du téléphone ou sur les applications en cours et qui met en pause la musique en conséquence (et qui la reprend lorsque l'application est de nouveau en premier plan).

## 9 Accessibilité

Nous avons fait le choix d'élargir le public visé par notre jeu aux daltoniens, aux malvoyants et aux dyspraxiques visuo-spatiaux, en respectant les règles d'accessibilité en matière de couleurs, de contraste et de retranscription vocale.

En effet, toutes nos associations de couleur respectent la norme **WCAG 2.0 AA** [6] pour leurs catégories d'utilisation, permettant aux personnes atteintes de daltonisme d'utiliser notre application sans souci.

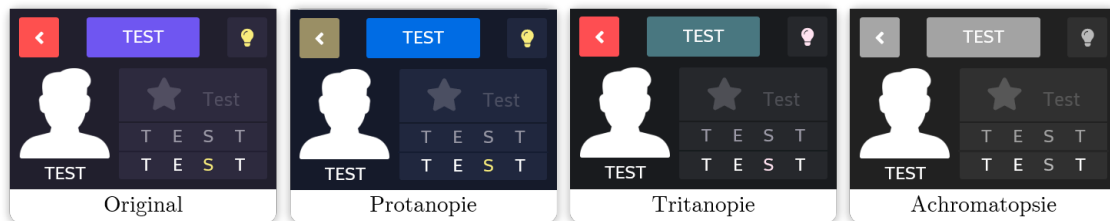


FIGURE 6 – Différence de couleurs et contrastes selon les formes de daltonisme

### 🔍 Qu'en est-il des malvoyants ?

Les icônes et images de notre application bénéficient d'une description alternative, permettant aux personnes souffrant de légers troubles de la vision (malvoyants, dyspraxiques visuo-spatiaux) d'utiliser des lecteurs d'écran afin de pouvoir jouer à notre jeu, étant donné que la seule image ne pouvant pas bénéficier d'une description est l'image à trouver, qui est assez grosse et contrastée pour être visible.

## 10 Autres fonctionnalités

Ci-dessous, une liste de petites fonctionnalités supplémentaires :


- Vibration du téléphone selon un tempo précis lorsque le joueur gagne un niveau (avec gestion des permissions).
- Changement de langue *in-app* sans redémarrage de l'application nécessaire.
- Affichage du popup de victoire de manière asynchrone et différée.
- Création d'une classe permettant d'implémenter les icônes de FontAwesome au projet.
- Formatage multiple de texte (différentes polices, tailles, couleurs dans un seul texte).
- Ajout de liens cliquables vers les sources dans les crédits.
- Copie en cache de l'objet *Level* courant afin de réduire les appels à la base de donnée.

## 11 Code source

Par habitude et étant donné que notre code source est disponible publiquement sur Github, nous nous sommes efforcés de produire un code propre et commenté (en anglais). Dans l'arborescence de notre projet, différents *packages* renferment les différentes parties de notre application. Parmi eux on notera les principaux : *menus* et *popups* où sont stockés tous les menus et popups de l'application, *utils* où sont stockés des classes facilitant l'intégration de fonctionnalités dans l'application et *classes* où sont stockés les autres classes.

En plus de la JavaDoc disponible en ligne [5], notre projet est entièrement commenté, et pour les parties qui peuvent être répétitives comme lors de l'application des couleurs et des icônes à la création des menus, nous avons séparés le code en plusieurs blocs visibles afin de retrouver facilement ces parties ou de passer directement au coeur logique du programme.

```

/*  VARIABLES

    • Retrieve views for design, click listener and/or gameplay
    • Retrieve other objects for gameplay purposes
*/

Button backButton = findViewById(R.id.backButton1);
TextView optionsText = findViewById(R.id.optionsText);
ImageView FRFlagImage = findViewById(R.id.FRFlagImage);
ImageView ENFlagImage = findViewById(R.id.ENFlagImage);
ImageView SPFlagImage = findViewById(R.id.SPFlagImage);
SeekBar volumeseekBar = findViewById(R.id.volumeSeekBar);
TextView copyrightText = findViewById(R.id.copyrightText);

```

FIGURE 7 – Exemple de bloc répétitif dans un menu : l'appel des vues

## 12 Conclusion

Watizit a été pour nous un moyen de s'entraîner par rapport à notre projet professionnel qui est de devenir développeur mobile, de mieux apprendre à travailler en équipe, d'acquérir de nouvelles connaissances, d'enrichir notre portfolio de projets et pourquoi pas de déployer notre application sur le marché après quelques mises à jour, car nous sommes assez satisfaits de l'avancement et du résultat de notre application en un temps imparti, et nous pensons qu'avec quelques efforts supplémentaires nous pourrions réaliser une meilleure version de l'application.

## Références

- [1] Documentation Android. <https://developer.android.com/docs>.
- [2] Flaticon. <https://www.flaticon.com>.
- [3] HorrorPen sur OpenGameArt. <https://opengameart.org/content/loop-lonely-witch>.
- [4] Enquête de l'INSV. [https://institut-sommeil-vigilance.org/wp-content/uploads/2019/02/DP\\_Journee\\_Sommeil2016.pdf](https://institut-sommeil-vigilance.org/wp-content/uploads/2019/02/DP_Journee_Sommeil2016.pdf).
- [5] JavaDoc du projet. <https://1visible.github.io/Watizit/index.html>.
- [6] WCAG 2.0. <https://www.w3.org/Translations/WCAG20-fr>.