

Rapport du projet Watizit

Nicolas Lauret, Andriniaina Randrianasolo

Université de la Réunion 2019-2020

17 mai 2020

Sommaire

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

Introduction

À qui s'adresse cette application ?

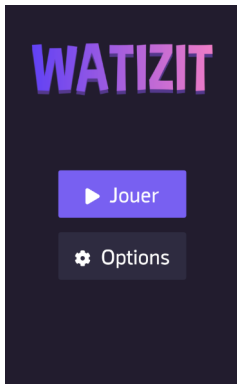
- Toute personne âgée de +10 ans qui veut se divertir
- Passionnés de jeux simples et rapides à prendre en main
- Compatible avec Android Jelly Bean (API 16) et plus
- Disponible en français, anglais et espagnol
- Accessible aux daltoniens, malvoyants et dyspraxiques

Plan

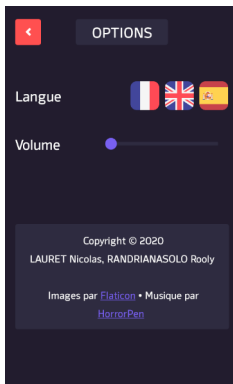
- 1 Introduction
- 2 Description générale
- 3 Organisation du travail
- 4 Design
- 5 Animations
- 6 Interactions et données
- 7 Gameplay
- 8 Musique
- 9 Accessibilité
- 10 Autres fonctionnalités
- 11 Code source
- 12 Conclusion

Description générale

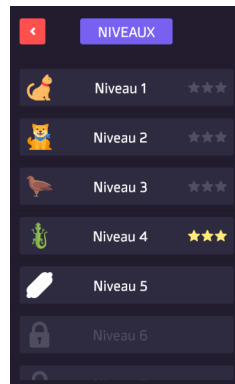
Les menus principaux



(a) Menu principal



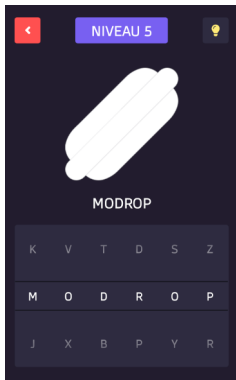
(b) Menu des options



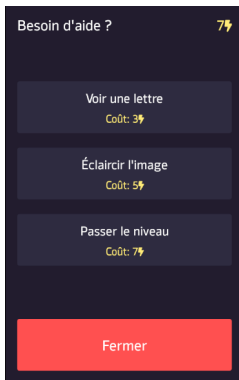
(c) Liste des niveaux

Description générale

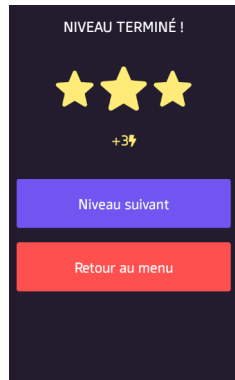
Composition du niveau



(a) Exemple de niveau



(b) Popup d'indices



(c) Popup de victoire

Plan

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

Organisation du travail

Mode de fonctionnement

- Phase de recherche
- Mise en place d'outils

Outils mis en place

- 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

Plan

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

Design

Évolution du design

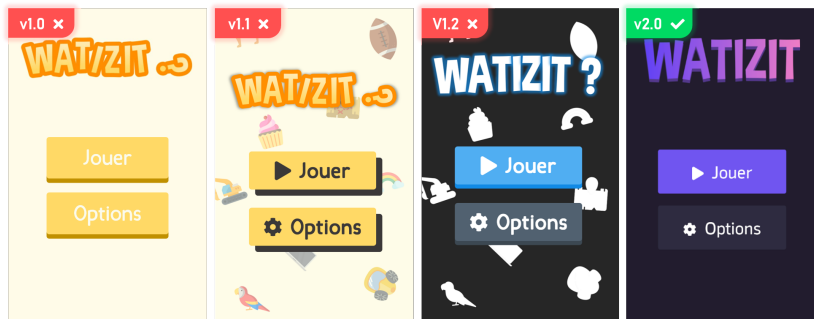


FIGURE – Évolution des designs réalisés sur Figma

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)
- 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

Plan

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

Animations

Boutons et images

Élaboration des animations

Pour rendre l'application plus vivante et dynamique, nous avons ajouté des animations d'apparition sur certains boutons ainsi que certaines images de notre application. Grâce à la formule ci-dessous :

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

Animations

Boutons et images

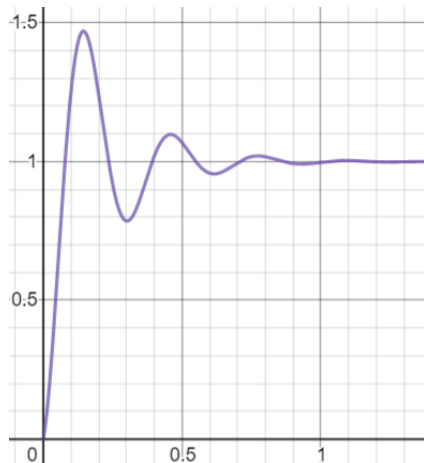


FIGURE – Représentation graphique de la fonction d'accélération

Animations

Transition de menus

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>
```

Plan

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

Interactions et données

Les interactions

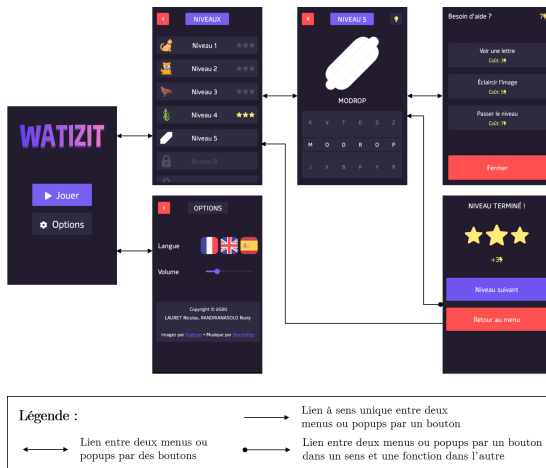


FIGURE – Organigramme des interactions entre les écrans de l'application

Interactions et données

Les interactions

Listing 2 – Interaction 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);  
}
```

Interactions et données

Les données

Quelles données sont manipulées et stockées ?

- La langue, le volume de la musique et la monnaie en jeu sont stockés dans les SharedPreferences.
- Les niveaux sont stockés dans une base de donnée.

Interactions et données

Les données

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 levels list activity, default=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; }
```

Interactions et données

Les données

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;
    }
}
```

Plan

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

Gameplay

Fonctionnalités de création

- Système de mélange de lettres uniques aléatoires
- Sécurité pour éviter que le mot apparaisse au début

Objectif et récompense

- Score en fonction du temps effectué
- Récompense sous forme d'éclairs (monnaie en jeu)

Gameplay

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

on a :

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

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 :
 $\text{tempsFinal} = 0 + 12000 \times 4 = 48000\text{ms} = 48\text{s}$, or pour gagner 3 étoiles il faut finir le niveau en moins de 45 secondes.

Plan

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

Implémentation de la musique de fond

- Création d'un *Service* contenant un *MediaPlayer* pour une musique globale
- Création d'un lien entre le *MediaPlayer*, le *Service*, les *SharedPreferences* et le *SeekBar* pour le volume
- Ajout d'un *HomeWatcher* pour l'écoute du bouton "Home"

$$f(\text{volume}) = 1 - \frac{\log(\text{max} - \text{min} + 1 - \text{volume})}{\log(\text{max} - \text{min} + 1)}$$

avec $\text{max} = 100$ et $\text{min} = 0$ dans notre cas

Plan

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

Accessibilité

Web Content Accessibility Guidelines

Qu'est ce que les WCAG 2.0 ?

C'est un ensemble de normes et de recommandations pour rendre les contenus plus accessibles.

Normes appliquées (WCAG 2.0 AA)

- Couleurs et contrastes accessibles aux daltoniens
- Description alternative des images et icônes pour les lecteurs d'écran

Accessibilité

Exemple de norme respectée

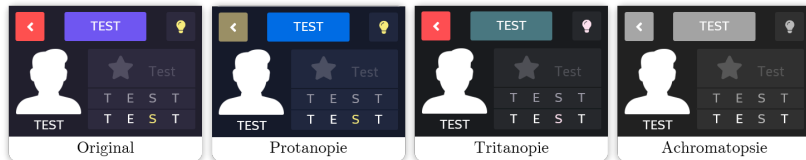


FIGURE – Différence de couleurs et contrastes selon les formes de daltonisme

Plan

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

Autres fonctionnalités

Petites fonctionnalités supplémentaires


- Vibration du téléphone lorsque le joueur gagne un niveau
- Changement de langue *in-app* sans redémarrage nécessaire
- Affichage de popup de manière asynchrone et différée
- Classe permettant d'implémenter les icônes de FontAwesome
- 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

Plan

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

Petites fonctionnalités supplémentaires

- Fichiers séparés par catégories dans des *packages*
- JavaDoc et commentaires détaillés en anglais
- Séparation du code répétitif en blocs

```
/*  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 – Exemple de bloc répétitif dans un menu : l'appel des vues

Plan

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

Conclusion

Que nous a apporté ce projet ?

- Entraînement par rapport à notre projet professionnel
- Nouvelle expérience de travail en équipe
- Acquisition de nouvelles connaissances
- Enrichissement de notre portfolio de projets
- Déploiement possible de l'application