Lespagnol Killian
Tribouillois Julien
12/04/2024

Rapport du projet d’Android

Lol Build

Table des matières

I. Objectif de l’application :

Cette application a été développé avec un SDK 27 au minimum et un SDK 34 en target. L’application a été testé et validé avec un émulateur de type *Pixel 3a API 27 (Android 8.1 (‘’Oreo”)*. Le but de cette application est de fournir un build pour le jeu League of Legends. Cette idée est survenue car nous jouons tous les deux à ce jeu et nous nous sommes rendu compte qu’une application spécialisée serait plus efficace que des recherches sur internet en même temps de jouer.

League of Legends est un jeu de stratégie dans lequel deux équipes de cinq joueurs s’affrontent pour détruire la base ennemie. Chaque joueur doit choisir son personnage et les objets à acheter pour vaincre ses adversaires.

II. Les étapes de réflexions et conceptions :

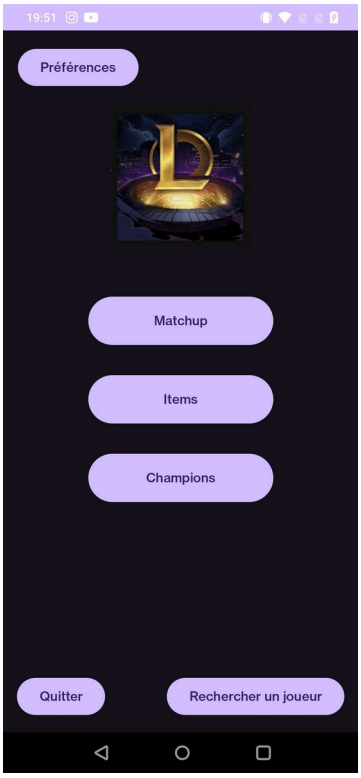
1. L’API Data Dragon et Riot API:

Pour ce Projet nous avons utilisé principalement l’API *Data Dragon* de Riot Games. Cette API permet d’avoir accès à toutes les données et les actifs du jeu League of Legends, comme les champions, les items et autre données utiles.

En ce qui concerne la *Riot API* elle plus utile pour retrouver des données comme les joueurs, leurs statistiques et leurs historiques de parties.

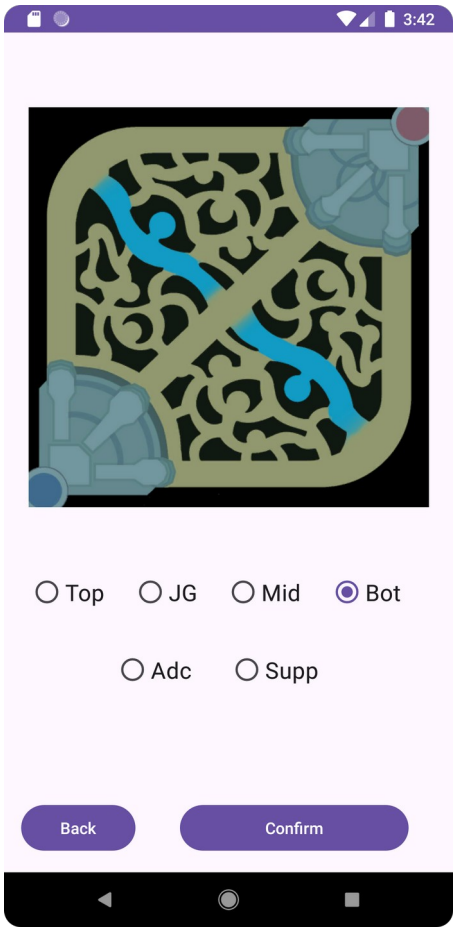
2. Vue générale de l’application :

La principale fonctionnalité de cette application est de proposer une liste d’items personnalisés, pour cela nous allons commencer par demander le poste où l’utilisateur jouera, puis nous lui demanderons son personnage ainsi que celui de son adversaire, après cela une page s’affiche avec les informations qu’il a besoin pour remporter sa partie. D’autres options sont disponibles sur cette application, on peut par exemple rechercher un item ou un champion en particulier, ou encore obtenir la liste des anciens match d’un joueur en indiquant son pseudo. Vous pourrez trouver en annexes les premières maquettes des activités que nous avons plus ou moins modifiés par la suite.

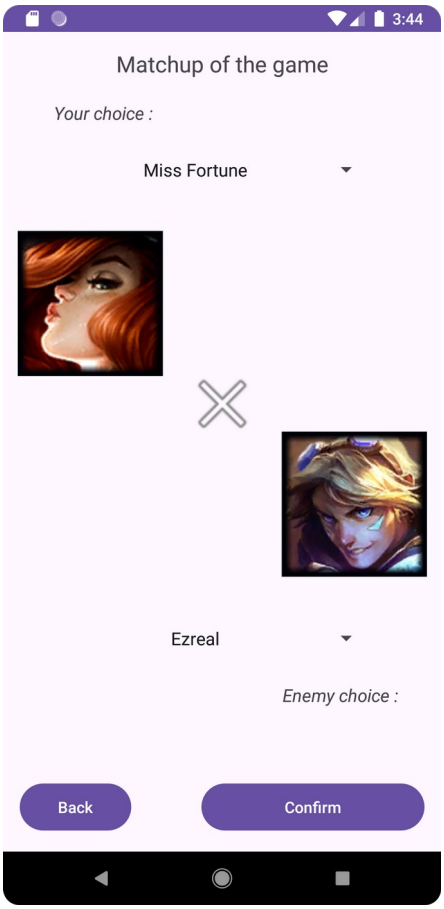


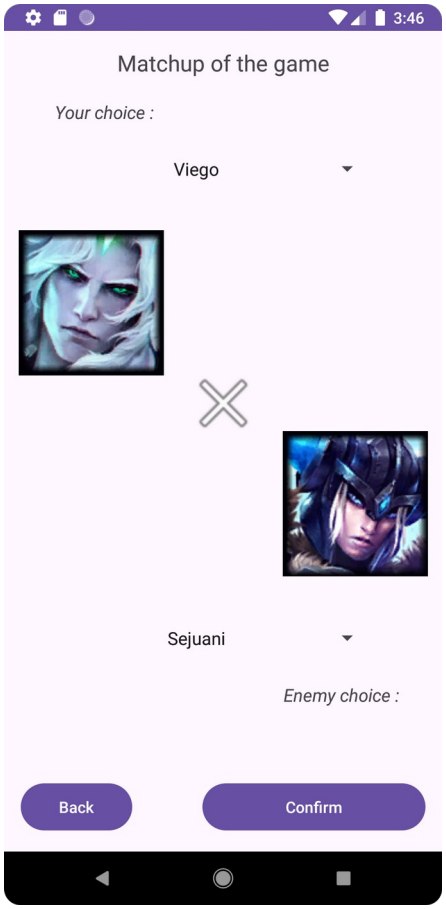
3. La construction du build optimisé :

La partie la plus importante de cette application est celle de trouver un build adapté à son adversaire, pour arriver à cet objectif nous devons passer par plusieurs étapes.

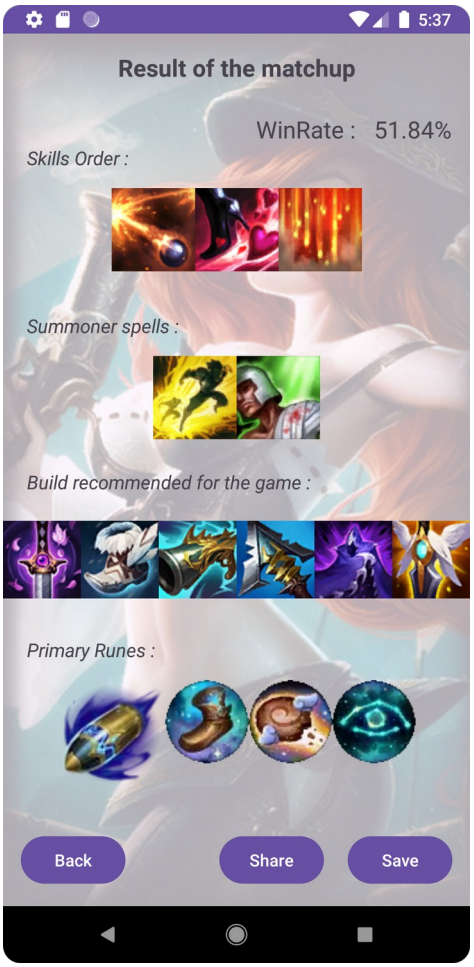


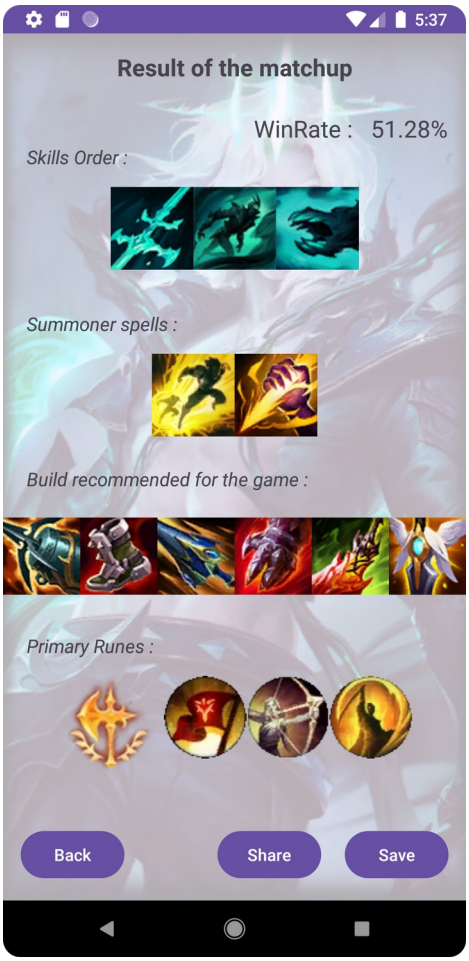
Tout d’abord nous avons le choix du rôle, en effet la partie et les adversaires seront totalement différents en fonction du post occupé par le joueur. Pour cela nous avons affiché la carte ainsi que les rôles avec des radios boutons, à noter que les rôles *Adc* et *Supp* sont cachés si le rôle *Bot* n’est pas sélectionné auparavant.





La seconde étape est celle du choix de son champion ainsi que celui de son adversaire, pour cela nous avons deux *Spinners* afin de faire défiler les personnages du jeu, après cette étape l'application doit récupérer les informations nécessaires pour afficher le *build* du personnage.

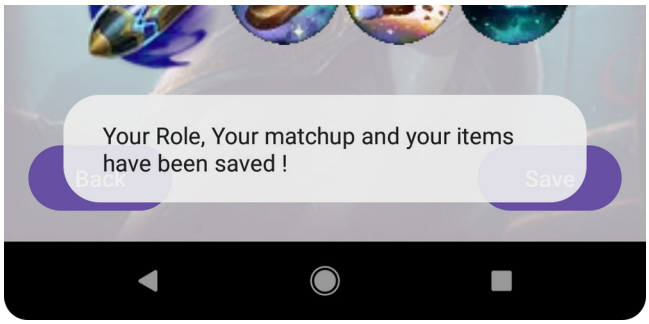




Après le chargement nous nous retrouvons sur la page de résultat ci-contre avec plusieurs informations. Tout d'abord nous avons le taux de victoire du matchup, ensuite nous avons l'ordre des compétences, puis les sorts d'invocateurs, ces derniers doivent être choisis par le joueur lors de la sélection du champion et vont changer sa manière de jouer. Ensuite nous retrouvons la liste d'objets dans l'ordre où les acheter pendant la partie, et les différentes runes à attribuer à son personnage lors de sa sélection.

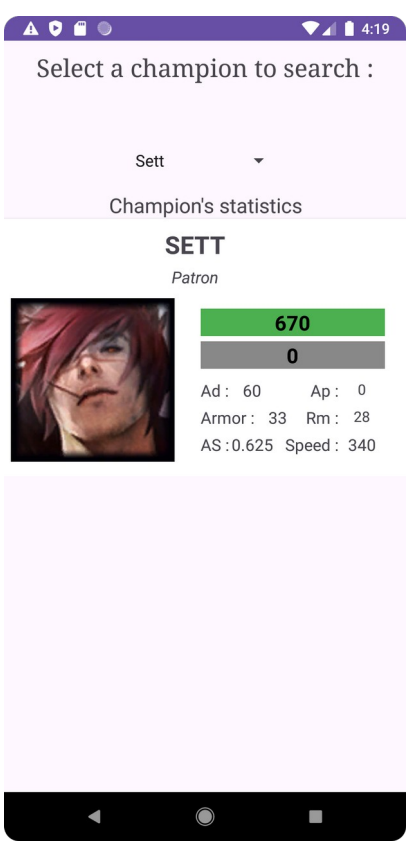
Un bouton *Share* nous permet d'envoyer le lien du site, ou nous avons récupérer le build, à l'utilisateur.

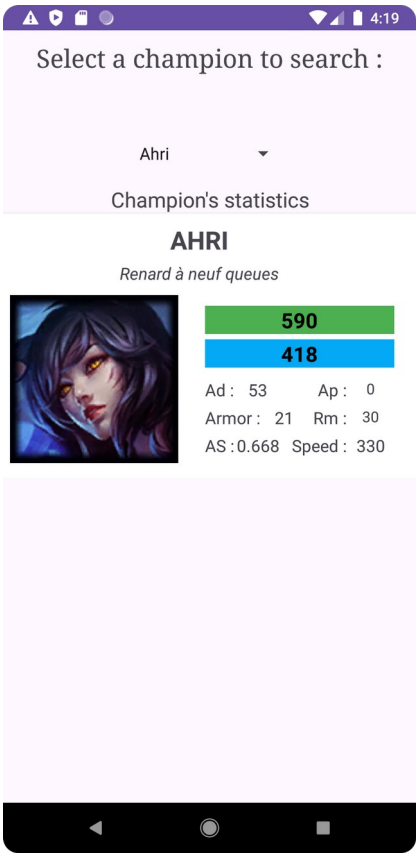
Un bouton *Save* en bas à droite nous permet de sauvegarder notre sélection dans une base de données afin de pouvoir l'afficher dans les préférences. Un message apparaît pour nous prévenir que la sauvegarde a bien été effectué.



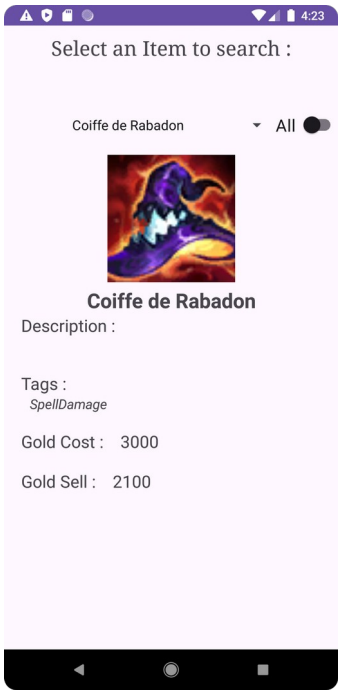
4. La recherche d’un item ou un champion :

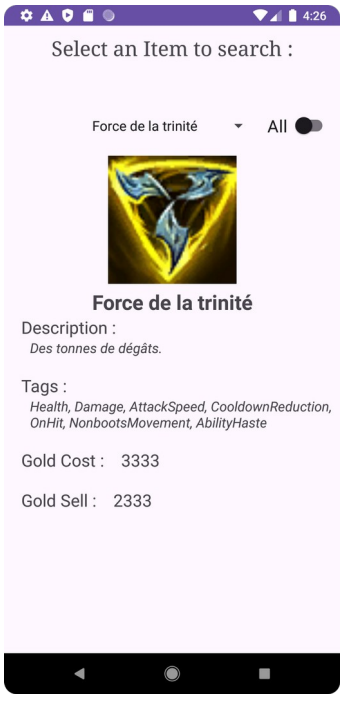
A partir de la page d’accueil l’utilisateur aura accès à d’autres fonctionnalités comme par exemple la possibilité de récupérer les informations d’un champion en particulier ou d’un item.





Remarque : Le changement de couleur de la seconde barre s’adapte selon le type de personnage choisi, certains possède du *Mana* alors que d’autre rien ou encore de la *Rage* (afficher en orange).





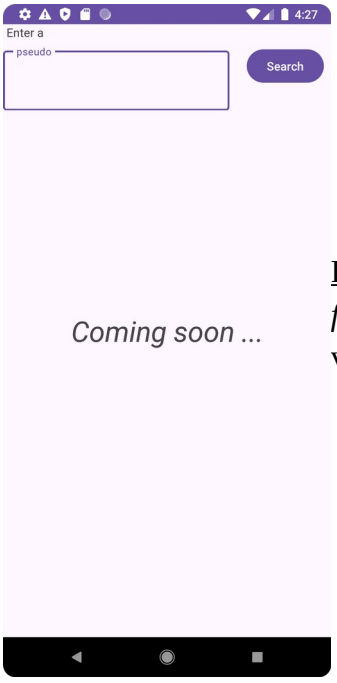
Après avoir cliquer sur le bouton *Champion* nous arrivons sur une page similaire aux deux premières captures, nous avons un *Spinner* afin de sélectionner le champion voulu, ensuite nous aurons les informations de base du champion tel que son nom, son titre, ses points de vie et de mana, ses dégâts physiques (Ad), ses dégâts magiques (Ap), son Armure, sa résistance magique (Mr), sa vitesse d'attaque (As) et sa vitesse de déplacement (Vitesse).

L'activité pour obtenir l'information sur un item fonctionne de la même manière que celle d'un champion. Après avoir sélectionné l'item voulu nous avons une légère description, ce qui est utile notamment si l'item a un effet passif ou actif. Le champs *Tag* permet d'identifier les caractéristiques que cela va modifier sur le personnage , par exemple ici la *coiffe de rabadon* va donner de la puissance magique. Nous pouvons aussi voir le coût original de l'item ainsi que son prix de vente. Le *Switch* «All» permet de choisir si on veut tous les objets ou seulement ceux qui sont présents actuellement dans le jeu.

5. La recherche des matchs précédents d'un joueur :

Depuis la page d'accueil nous voulions aussi rajouter une fonctionnalité permettant de rechercher les anciennes parties d'un joueur grâce à son pseudo. Cette partie n'est pas essentiel mais il est agréable de pouvoir voir les résultats des anciennes parties d'un adversaire ou même d'un allié, afin de voir si il à l'habitude de jouer sur son personnage par exemple.

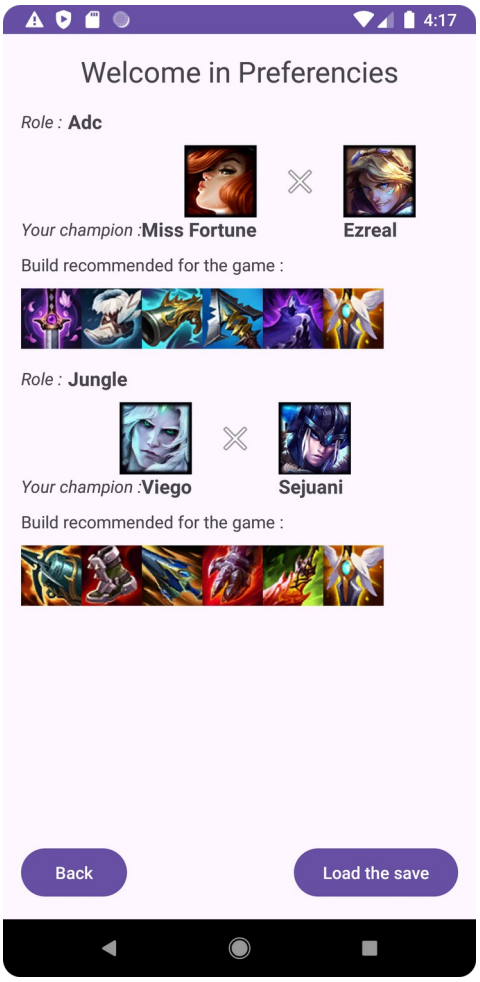
Après avoir sélectionner le pseudo d'un joueur et cliquer sur le bouton recherche, la liste des anciens matchs s'affichent dans un recycler view, une cellule du recycler view contient les informations principales d'un match : l'icone de son champion, la liste de ses runes, ses deux sorts d'invocateurs, le mode de jeu pour la partie, la date et la durée de la partie. Elle contient aussi son kda c'est à dire son score d'ennemie tué, de mort et d'assistance pour vaincre un ennemie, ses items ainsi que les champions présent dans la partie sont aussi affiché. Cependant par manque de temps nous n'avons pas fini d'implémenter cette fonctionnalité.



Remarque : Comme vous pouvez le voir c'est une future *feature* qui sera développé à l'avenir. Cette idée nous est venu au abord de la fin du projet.

6. Les préférences :

L'activité *Préférencies* nous sert d'affichage pour les données sauvegardées par l'utilisateur. A l'aide du bouton *Load the save*, nous affichons le contenu de la base de données dans un *Recycler View* sous forme d'item, avec *l'activité item_database* pour la partie *xml* et *Database_Adapter* pour la classe Java.



III. Les difficultés rencontrées et leur solution :

La principale difficulté et celle qui nous a fait perdre le plus de temps est l'utilisation d'une mauvaise API, en effet nous avions choisi d'utiliser l'API Riot API au début, cependant nous n'avons pas réussi à l'utiliser correctement. Après plusieurs recherches nous avons trouver une bibliothèque de Riot Games nommée *Orianna* pour la gestion des API en langage java, mais de nouveaux problèmes de communication avec les différentes API Riot sont survenus. Finalement nous avons fini par trouver l'API *Data Dragon* et à l'utiliser convenablement pour notre implémentation des champions et des items et de nombreuses autres fonctionnalités.

IV. Conclusions :

En conclusion ce projet fut très enrichissant, ce fut agréable de travailler sur un sujet qui nous tenais à cœur. Nous pensons d'ailleurs continuer à améliorer l'application dans le future, en effet énormément d'informations et d'optimisations peuvent être apportées pour qu'elle soit aussi efficace que les sites de build trouvables sur internet.

V. Technologies utilisées :

Pour ce projet nous avons utilisé différentes dépendances pour les activités de notre application. Concernant la gestion des appels d'API nous avons utilisé les dépendances de *Gson* pour la récupération et le traitement des requêtes d'API de Data Dragon ou encore de Riot API.

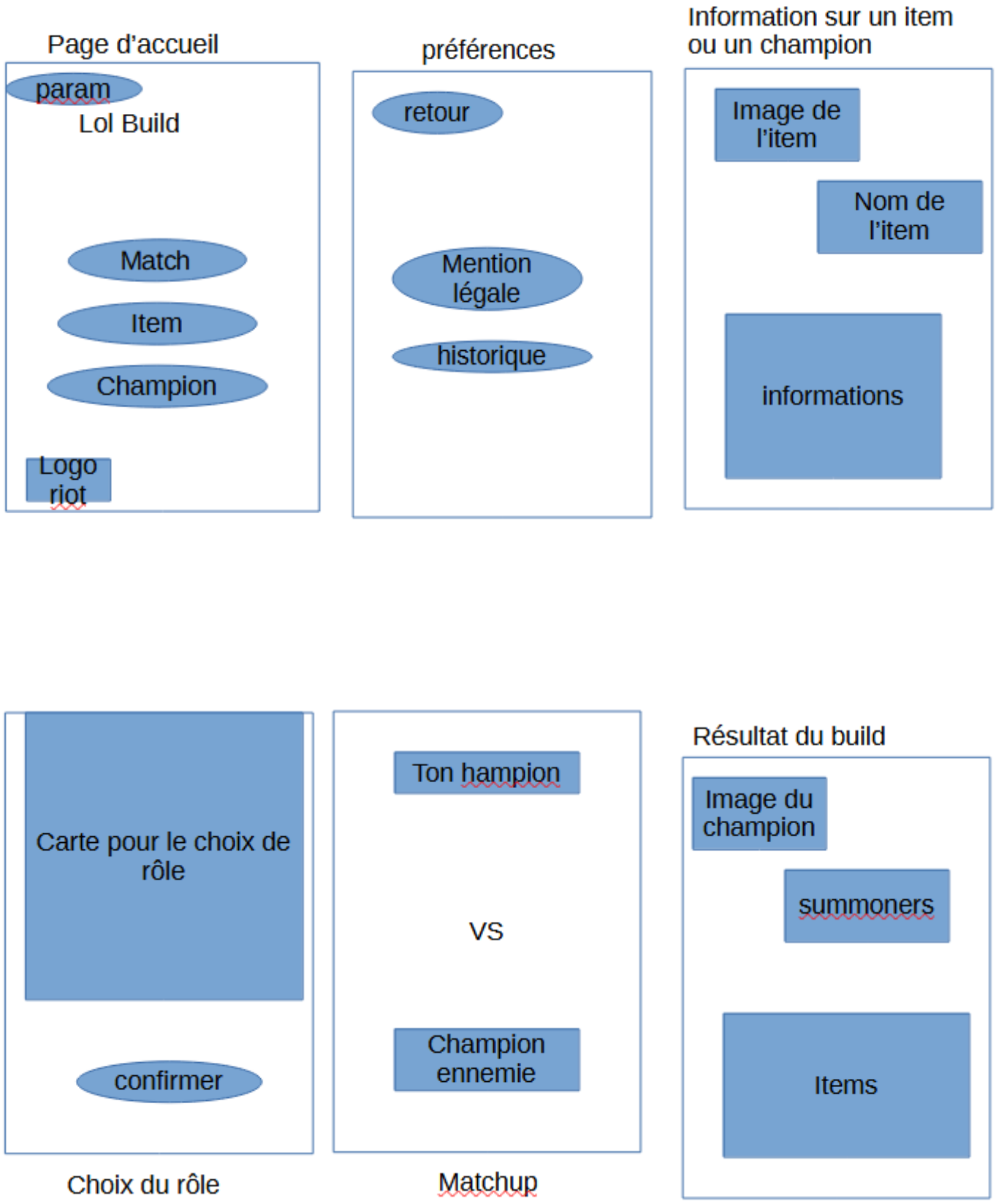
Concernant le *scraping web* nous avons utilisé *Jsoup* pour certaines informations de la page *ResultBuild*.

Tout ce qui concerne les images nous avons choisi les dépendances de *Coil* pour les téléchargements et modifications des images venant des API.

Pour finir nous avons utilisé la gestion de database d'Android avec le s dépendances de *Room* pour la partie stockage et l'affichage dans *Preferencies.*

```
    implementation("com.google.code.gson:gson:2.10")
    implementation("io.coil-kt:coil:2.6.0")
    implementation("org.jsoup:jsoup:1.14.3")
//Management of the Database Room used
implementation("androidx.room:room-runtime:2.6.1")
annotationProcessor("androidx.room:room-compiler:2.6.1")
```

VI. Annexes :


Page d'accueil
param
Lol Build
Match
Item
Champion
Logo riot
préférences
retour
Mention légale
historique
Information sur un item ou un champion
Image de l'item
Nom de l'item
informations
Carte pour le choix de rôle
confirmer
Choix du rôle
Ton hampion
VS
Champion ennemie
Matchup
Résultat du build
Image du champion
summoners
Items