Game of Life

Client: Pierre MAHIEUX

Développeur : Bastien BUREAU 26/01/2024

Table des matières

[I.) La demande client – Le jeu de la vie 2](#_Toc160718346)

[II.) Expression du besoin 2](#_Toc160718347)

[1.) Découpages des fonctionnalités et chiffrage 3](#_Toc160718348)

[2.) Scénario de test 4](#_Toc160718349)

[3.) Spécificité Techniques 4](#_Toc160718350)

[III.) Monétisation et rentabilité de l’application 6](#_Toc160718351)

[IV.) Lancement d’une session de jeu – Explication 6](#_Toc160718352)

[1.) Installation du jeu 6](#_Toc160718353)

[2.) Configuration de l’environnement de jeu 6](#_Toc160718354)

[3.) Explication du jeu 6](#_Toc160718355)

# La demande client – Le jeu de la vie

Le jeu de la vie est un exercice de programmation assez répandues, ou le but du jeu est de laisser « vivre » des cellules en autonomies selon certaines règles spécifiques, les voici :

* Une cellule morte possédant exactement trois cellules voisines vivantes devient vivante (elle naît) ;
* Une cellule vivante possédant deux ou trois cellules voisines vivantes le reste, sinon elle meurt.

Le client à besoin d’effectuer des simulations en appliquant différente règle. Donc pour répondre à ce besoin nous allons développer une application logicielle.

# Expression du besoin

Le client souhaite implémenter les fonctionnalités suivantes :

Pour la 1ère version :

* Des profils utilisateurs
* Des règles spécifiques à chaque utilisateur, par exemple : 3A2S (3 Alives, 2 Stay)
* L’algorithmie du jeu de la vie

Pour la 2ème version :

* Un tchat général pour que les joueurs puissent discuter entre eux
* Relancer une simulation tout en partageant la règle de l’utilisateur qui demande à rejouer
* Permettre de modifier la règle en éditant le champ prévu à cet effet

Actuellement il reste encore des fonctionnalités à développer comme la gestion et le traitement du mot de passe ainsi que son cryptage

## Découpages des fonctionnalités et chiffrage

Voici le diagramme de Gantt qui prends en compte le découpage des fonctionnalités :

Une image contenant texte, capture d’écran, carré, Caractère coloré

Description générée automatiquement

Total : 32 jours

Après calcul la TJM estimée est de 632 € HT.

Ainsi 632 \* 32 = 14 400€ le prix de la mission.

## Scénario de test

Dans ce chapitre nous allons définir différents scénarii de test en fonctions des fonctionnalités.

Pour le profil utilisateur :

Test Unitaire :

* L’utilisateur peut s’inscrire, et récupérer sa règle qui lui est associée.
* L’utilisateur peut mettre à jour son pseudonyme ou sa règle
* L’utilisateur peut relancer le jeu en partageant sa règle avec les autres joueurs

Test Visuel :

* L’utilisateur doit avoir un mot de passe assez sécurisé et doit apparaître encrypter dans la base de données.
* L’utilisateur peut envoyer un message dans le tchat et être vu par tout le monde
* L’utilisateur peut mettre en pause le jeu

Pour le jeu :

* Le jeu doit être différents en fonction de la règle (c’est-à-dire qu’au bout d’un nombre de génération courts ou élever) il n’y aura plus de cellule.

## Spécificité Techniques

La cible, pour ce projet, sont les applications logicielles. Les différents choix pour le développement de ce besoin sont multiples.

Pour un développement d’applications logicielles nous avons deux choix principaux de langage entre le Java et le C#, sachant que c’est une application qui doit être ergonomique pour l’utilisateur.

Étant les langages majoritairement déployés dans l’industrie on va utiliser soit le Java soit le C#.

En Java nous avons une librairie JavaFX qui permet de faire de l’UI et en C# nous avons le WPF est déjà présent nativement.

Les particularités entre les deux sont :

|  |  |  |
| --- | --- | --- |
| Critères | JavaFX | WPF |
| Langage | Java | C# (XAML pour la conception de l'interface utilisateur) |
| Plate-forme | Multiplateforme (Windows, macOS, Linux) | Multiplateforme avec les nouvelles versions de .Net |
| Licence | Open source (GNU General Public License v2 avec classe d'exception de recherche de pièces jointes) | Propriétaire (Inclus dans le framework .NET) |
| Intégration | Intégration native avec Java | Intégration native avec .NET |
| Outils de développement | Principalement basé sur IntelliJ IDEA, Eclipse, NetBeans | Visual Studio |
| Performance | Performant, mais peut-être moins rapide que WPF dans certains cas | Réputé pour ses performances élevées |
| Look and feel | Peut varier en fonction de la plate-forme | Intégration étroite avec le look and feel de Windows |
| Richesse des contrôles | Large gamme de contrôles, bien qu'un peu moins que WPF | Vaste bibliothèque de contrôles intégrés |
| Communauté | Communauté active mais plus petite que WPF | Grande communauté avec de nombreux tutoriels et ressources |
| Déploiement | Peut nécessiter l'installation de la machine virtuelle Java | Dépend de la version du framework .NET installée sur l'utilisateur final |
| Accessibilité | Possibilité de créer des applications accessibles, mais cela peut nécessiter plus d'efforts | Dispose de fonctionnalités intégrées pour l'accessibilité |
| Animation et graphiques | Capacités d'animation et de graphiques intégrées | Excellente prise en charge des animations et des graphiques |
| Écosystème | Riche écosystème Java, avec de nombreuses bibliothèques tierces | Écosystème .NET robuste avec une large gamme de bibliothèques tierces |

Pour un jeu de la vie qui génère beaucoup de cellule en peu de temps il faut qu’on soit assez robuste sur les performances et qu’on a une bonne ergonomie graphique.

De plus j’ai une bonne expérience en C# ce qui nous permet un gain de temps car je n’ai pas besoin d’apprendre un nouveau langage.

Ainsi le choix s’est porté sur le WPF qui nous permettrait de respecter ces deux conditions de manière plus simple que pourrait le faire JavaFX.

Windows est dominant dans le marché d’utilisation des systèmes d’exploitation pour les particuliers. Afin de rester dans le même type de charte graphique afin de ne pas perdre l’utilisateur. Cela confirme donc notre choix.

Une image contenant texte, capture d’écran, Police, Bleu électrique

Description générée automatiquement

Source : <https://www.leptidigital.fr/technologie/parts-marche-systemes-exploitation-26774/>

# Monétisation et rentabilité de l’application

Sachant que l’application est en libre d’accès, on pourrait proposer aux utilisateurs :

* Des thèmes payant pour changer la couleur de leur profil / jeu
* Des thèmes de « board » pour le jeu de la vie
* Des skins pour les cellules afin de les améliorer et les rendre plus ergonomiques
* L’incrustation de pub dans notre logiciel.
* Proposer un abonnement pour ne plus avoir de pub.

# Lancement d’une session de jeu – Explication

## Installation du jeu

Sur GitHub on peut télécharger la release avec le tag V1.0.0 pour le server et le jeu.

Le docker de la base de données est avec le jeu.

## Configuration de l’environnement de jeu

Un docker compose est disponible pour le lancement du projet. Ce docker va permettre d’avoir la base de données les logins pour la base de donnée sont ID : root, mdp : root.

Voici la commande pour lancer le docker (une fois que vous êtes dans le dossier contenant le projet) docker compose up.

Il faut aussi lancer le serveur télécharger.

Une fois le docker lancé vous pouvez lancer autant d’instance du jeu que vous souhaitez.

## Explication du jeu

Une fois les conteneurs lancés vous pouvez lancer une instance du jeu de la vie.

* Une cellule peut naître, si cette cellule est considérée comme morte et si elle possède trois cellules voisines vivantes.
* Une cellule reste vivante uniquement si elle possède deux ou trois voisines vivantes.
* Une cellule meurt si elle possède moins de deux voisines vivantes.