Cahier des charges

Par : L'équipe TZ

Université de Strasbourg

Projet Intégrateur 2020-2021

L3 Informatique

Développement du jeu CodeNames

Table des matières :

Historique des modifications et révisions de ce document

N° de version	Date	Description et circonstances de la modification
V 0	27/01/2021	Brouillon : première version suite à la réunion initiale du groupe
V 1	31/01/2021	Version proposée au tuteur pour approbation
V 1.1	02/02/2021	Modification de quelques parties suite à la réunion avec le tuteur du groupe

1. CodeNames[1]

1.1 Présentation Générale

CodeNames est un jeu de cartes et d'association d'idées conçu par Vladimír Chvátil en 2015 et publié par Czech Games Edition. Il est édité en France par Iello. Pour une expérience de jeu optimale, ce jeu est recommandé pour au moins quatre personnes. Cependant, il peut être joué par deux joueurs et jusqu'à huit joueurs. CodeNames a fait sensation partout dans le monde depuis sa sortie. Jusqu'à présent, il a été publié en trente-huit langues, comprenant six alphabets différents.

1.2 Principe du jeu

Les joueurs sont répartis en deux équipes : rouge et bleue. Chaque équipe est composée d'un Maître-Espion et des Agents en mission. Le but du jeu est de prendre contact avec ses informateurs tout en évitant de retrouver les informateurs ennemis mais surtout, l'assassin caché.
Pour cela, les Maîtres-Espions donnent, tour à tour, un et un seul mot d'indice pouvant désigner plusieurs Noms de Code sur la table. Leurs Agents essaient, dès lors, de deviner les Noms de Code de leur couleur et d'éviter ceux de l'autre équipe. Et par-dessus tout, ils doivent éviter l'Assassin !
L'équipe qui identifie tous ses informateurs en premier, sans contacter l'assassin, gagne la partie.

1.3 Règles du jeu

Vingt-cinq cartes de Noms de Code (CodeNames), chacune portant un mot, sont disposées dans une grille rectangulaire de 5×5, dans un ordre aléatoire. Un certain nombre de ces mots représentent des agents rouges, un autre nombre représente des agents bleus, un mot représente un assassin, et les autres représentent des passants innocents. Les Maîtres-Espions ont accès à la carte qui indique quels mots correspondent à leur équipe et qu'ils doivent donc faire deviner.
À chaque tour, le Maître-Espion de l'équipe donne un indice constitué d'un seul mot (qui ne figure sur aucune des vingt-cinq cartes), ainsi que le nombre de mots qu'il pense faire deviner avec cet indice.
Le mot donné doit être lié à autant de mots que possible sur les cartes de ses propres agents, mais pas à d'autres, de peur qu'ils ne conduisent accidentellement leur équipe à choisir une carte représentant un spectateur innocent, un agent ennemi, voire l'assassin.
Le nombre donné en indice indique aux agents combien de cartes sont liées au mot de l'indice. Il détermine également le nombre maximum de devinettes que les agents peuvent faire au cours de ce tour ; ce nombre est égal au nombre en indice plus un. Les agents doivent faire au moins une proposition par tour, au risque de se tromper et d'en subir les conséquences. Ils peuvent également mettre volontairement fin à leur tour à tout moment par la suite.

- si le mot désigné est bien un mot à deviner, le mot est recouvert des couleurs de l'équipe ;
- si le mot désigné appartient à l'autre équipe, il est recouvert des couleurs de l'équipe adverse ;
- si le mot désigné n'appartient à aucune équipe, on passe au tour de l'équipe suivante ;

[1] Wikipédia, iello.fr

- si le mot désigné est le mot « assassin », l'équipe en question perd immédiatement la partie.

La partie se termine si une équipe arrive à identifier tous ses agents et donc elle gagne la partie, ou bien si une équipe retrouve l'assassin, dans ce cas, elle perd la partie.

1.4 Réception et popularité

Le jeu a atteint la dix-septième position (sur plus de 84000) dans le fameux classement BoardGameGeek des jeux de tous les temps, et la première position dans le classement BGG des jeux de société. En 2016, Il remporte le Spiel des Jahres (Jeu de l'année), la récompense la plus prestigieuse au monde pour un jeu. CodeNames a également remporté le prix Origins du meilleur jeu familial, du jeu familial préféré des fans et du jeu de l'année 2015.

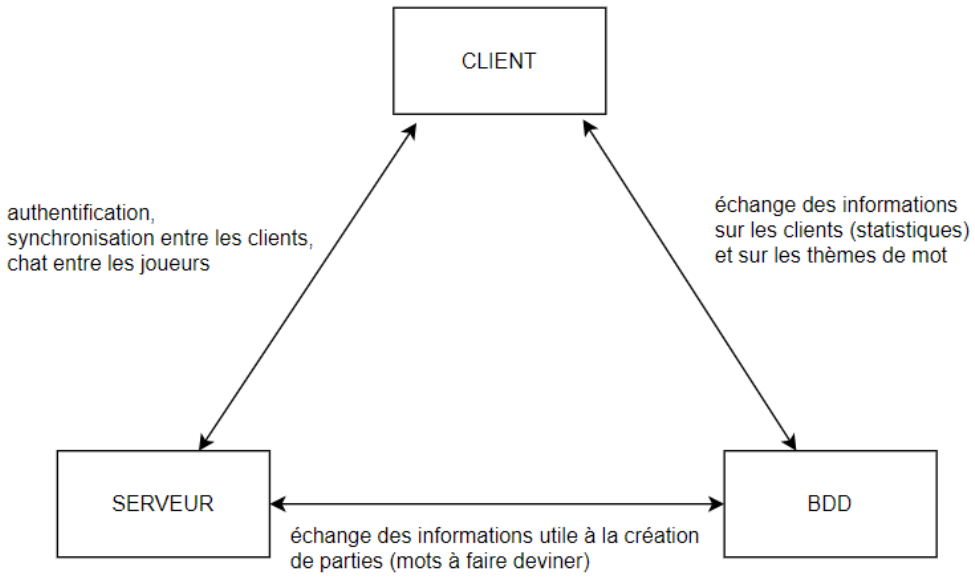
2. Architecture du projet

2.1 Eléments clés de la programmation

POO:
Création du client (interface du jeu et des serveurs, règles, contraintes et mécanismes de jeu).

BDD:
Communication des informations avec le client pour l'archivage des statistiques et l'envoi des thèmes de mots au serveur de jeu pour qu'il constitue une liste aléatoire.

RES:
Connexion du Client vers les serveurs de jeu, gestions des chats et communication des différents serveurs disponibles vers le client.



2.2 Présentation des équipes

POO :

- **POO-BE :**
  Cette équipe sera chargée de la partie Back End du projet. Elle aura comme rôle d'implémenter les différentes fonctionnalités et mécanismes internes afin d'assurer le bon déroulement du jeu.
  L'équipe POO-BE est composée de : N.WEISS et P.WEILER.
- **POO-FE :**
  Cette équipe s'occupera de la partie Front End. Elle sera chargée de créer le graphisme du jeu et toutes les interfaces nécessaires ainsi que gérer l'animation et l'affichage.
  L'équipe POO-FE est composée de : L.WEREY et Y.TALY.

BDD :

Cette équipe aura comme rôle de créer une base de données complète qui répondra aux exigences et contraintes du jeu.
L'équipe BDD est composée de : T.JUBA

RES :

- **RES-SC :**
  Cette équipe sera dédiée à la création de différents mécanismes nécessaires pour l'interaction des clients avec le jeu.
  L'équipe RES-SC est composée de : C.XU et S.ZHANG

- **RES-SBDD :**
  Le rôle de cette équipe sera la conception et l'implémentation des mécanismes assurant la communication entre le jeu et la base de données principale.
  L'équipe RES-SBDD est composée de : T.BOAN et T.Huayi.

2.3 Liste des règles et fonctionnalités proposées

Index de coloration : Priorité haute ; Priorité moyenne ; Priorité faible

CONNEXION MULTIJOUEUR (session random ou avec amis) :

- Salon de connexion avec la possibilité de choisir les rôles

- Browser de salon -> Possibilité de filtrer les salons (nom de la partie, thème)

- Salons privés (nécessite un mdp pour y accéder)

- Pseudo unique -> On ne peut pas avoir deux pseudos pareils

- Stats du joueur (nombre de parties gagnés, pourcentage de bonnes réponses, classement générale)

PARAMETRAGE PARTIE

- Option de personnaliser le nombre max & min des joueurs

- Langage des mots : autres langues que FR & EN

- Mots personnalisés -> éditeur de thèmes

- Temps de réponse pour les espions (Chronomètre)

- Nombre de carte dans la partie (autre que 5*5)

- Thèmes de mots

- Nombre d'espions > 1

- La gestion des joueurs qui se connectent en cours d'une partie

- Chat vocal

-Choix aléatoire rôles et équipes

AFFICHAGE PARTIE

- Affichage des pseudos des joueurs, leurs rôles et leurs équipes

- Chat entre les joueurs -> (Un chat réservé aux espions ?)

- Les agents voient juste les cartes avec les mots de couleur neutre à part pour ceux déjà devinés

- Les espions voient toutes les cartes rouges, bleues, grises et noires

- Compteur de mots restants à deviner (= Score)

- Historique des actions

- Boutons pour recommencer et quitter la partie

DEBUT PARTIE

- Au moins un espion et un agent dans chaque équipe

- Choix de l'équipe qui commence : au hasard

- L'équipe qui commence a toujours un mot en plus à deviner

- Mots aléatoires en fonction des thèmes choisis, avec les couleurs

DEROULEMENT PARTIE

- Au début du tour l'espion donne un mot et un seul (ainsi que le nombre de cartes à faire deviner) qui s'affiche pour tout le monde

- Les agents peuvent marquer des mots quand c'est leur tour

- L'équipe devine le nombre de mots qu'elle veut jusqu'à ce qu'un mot ne soit pas de leur couleur -> affichage de la couleur de la carte

- A partir de 1 carte retournée, l'équipe peut décider de finir son tour -> bouton ‘Finir tour’

- Fin de partie quand le compteur d'une équipe tombe à 0 -> animation de victoire

- Restriction du chat pour les espions

- Restriction sur l’indice donné + pénalité

THEMES

- Large set de mots suivant un thème défini en deux langues (ANGLAIS, FRANCAIS)

- Editeur de thème

3. Choix techniques

3.1 POO

Choix de technologies :

Type de technologie	Choix final	Autres choix considérés	Justification du choix
Back End	Unity (C#)	Java	C’est un outil spécialement conçu pour la création de jeux vidéo qui nous permettra de faire la transition de notre jeu vers le web plus facilement.
Front End	Unity	Java	Comme c’est un moteur de jeu populaire, il y a beaucoup de ressources en ligne concernant la création de jeu en ligne. Unity permet aussi un export vers une application web simplifiée. Il est bien adapté pour l'intégration avec le réseau. Il permet une implémentation du graphisme 3D et de l’interface, ce qui serait plus difficile avec JAVA.

3.2 BDD

Choix de technologies :

Type de technologie	Choix final	Autres choix considérés	Justification du choix
Gestion de BDD	PHPMy Admin	Navicat	Pratique, car c'est l'outil de gestion MySQL basé sur le Web le plus utilisé

La BDD	MySQL	Aucun	1. Tous les membres ont des connaissances de base pertinentes, d'excellentes performances et un service stable 2. La communauté et les utilisateurs sont très actifs, lorsque nous rencontrons des problèmes, nous pouvons obtenir de l'aide rapidement. 3. Facile à entretenir, faibles coûts d'installation et de maintenance. 4. Compatibilité : prend en charge plusieurs systèmes d'exploitation, fournit plusieurs ports API et prend en charge plusieurs langages de développement.

3.3 RES

3.3.1 Serveur Client

Choix de technologies :

Type de technologie	Choix final	Autres choix considérés	Justification du choix
Langage d'implémentation	Java	C++	Framework excellent existant, qualité du code et qualité d'architecture du programme bien garanties. Assez puissant pour des calculs qui ne sont pas trop lourds.
Définition du format de paquet de message	Protocol buffers by Google Inc.	JSON	Facile à démarrer, portabilité pour des plateformes et des langages différents.
Data Persistance	Hibernate	MyBatis	Convenance de développement et configurations (Intégré dans Spring JPA). On n'a pas de besoin fort pour la personnalisation des requêtes de BDD, donc l'avantage de flexibilité de MyBatis ne sera pas évident.
Authentification	Spring Security	Aucun autre choix ne répond au besoin aussi bien que Spring Security	Compatibilité parfaite avec Spring Framework, la gestion de droit au niveau de méthode rencontre exactement le besoin d'un serveur de type RPC. Intégration des gestions de l'authentification et de l'autorisation.
Proxy inverse	Nginx	Apache	Nginx a été conçu pour être un proxy inverse. Apache peut parfaitement inverser le proxy,

			mais il nécessite plus de ressources que Nginx

3.3.2 Serveur BDD MySQL

Choix de technologies :

Type de technologie	Choix final	Autres choix considérés	Justification du choix
Construction du Serveur BDD sur nos propre ordinateur	Wampserver	Xampp	Inclue Apache web server, PHP interpréteur et MySQL, ce qui permettra au groupe d'avoir le même environnement de test et gagner plus de temps pour le développement.

Au début, l'équipe a considéré comme options soit une base de données SQL (relationnelle), soit une base de données NOSQL (non relationnelle). Cela dépend de la conception de structure des données et du niveau de cohérence des données que l'équipe de BDD aurait choisi : si la structure des données n'est pas fixe ou change fréquemment, NOSQL sera un bon choix. En outre, si l'équipe BDD avait choisi un haut niveau de cohérence des données, SQL aurait été le meilleur choix. Finalement, l'équipe a choisi **SQL**.

Pour le choix du langage responsable de l'interaction entre la partie serveur et BDD, l'équipe a choisi Java. C'est un langage que tous les membres ont déjà utilisé et il sera aussi utilisé par l'équipe Front End pour assurer la compatibilité entre les deux groupes.

Pour la construction du serveur, il y avait 3 choix :

Langage	Avantages	Inconvénients
MySQL	- Multi-thread, multi-utilisateurs - BDD relationnelle - Authentification basée sur l'hôte. - Disponible même sans Internet - Capable de fournir un serveur en tant que programme autonome pour l'environnement réseau client / serveur	- Lorsque la quantité de données atteint une certaine échelle, afin de maintenir la cohérence, elle est très sujette à des problèmes de concurrence, tels que des blocages, de sorte que la vitesse de lecture et d'écriture diminue très sérieusement - Requête dans une table contenant une grande quantité de données, l'efficacité est très faible
Oracle	- Compatible avec IBM SQL / DS, DB2, INGRES, IDMS / R, etc. - BDD relationnelle - Il peut fonctionner sous différents systèmes, tels que VMS, DOS, UNIX et Windows.	Presque pareil que ceux de MySQL

MongoBD	- Face au document - BDD non relationnelle - Pas besoin de passer par l'analyse de la couche SQL, performances de lecture et d'écriture élevées. - Sur la base de paires clé-valeur, les données ne sont pas couplées et sont donc faciles à développer - Formats de stockage multiples : prennent en charge le format clé-valeur, le format de document et le format d'image, tandis que les bases de données relationnelles ne prennent en charge que les types de base - Un bon choix de serveur de cache	- L'avantage est au détriment du principe ACID : bien que cette solution augmente considérablement le temps disponible et l'évolutivité, elle peut également provoquer une perte de données -MongoDB ne soutient pas les opérations de jointure et il n'est pas recommandé aux applications qui nécessitent des requêtes complexes.

4. Vision du rôle du chef de projet

Pour ce projet, les tâches d'un chef de projet consisteront à :

- piloter le projet et gérer les cycles de développement
- assurer le respect des délais fixés et gérer les retards en fonction des priorités
- coordonner entre les différentes équipes de développement
- suivre et évaluer le progrès régulièrement
- assurer l'intégration continue
- faire un reviewing du code
- agir comme un lien entre l'équipe et le tuteur en lui donnant une vision claire du travail effectué
- rédiger les comptes rendus hebdomadaires du progrès effectué ainsi que les rapports des réunions de mardi.

5. Tâches et équipes

5.1 POO : Tâches et timeline

5.1.1 POO-BE

Liste des tâches principales :

Bases :

- Création des cartes (objet)
- Création des joueurs (objet)

Pré-jeu :

- Vérification du pseudo (s'il n'existe pas déjà)
- Vérification du mot de passe
- Vérification du nombre de joueurs min/max d'un salon (griser le salon si c'est déjà max ou empêcher le clic, etc.)

Jeu :

- Début :
  - Génération des cartes sur le plateau
  - Récupération des données (mots, stats) dans la BDD
  - Nombre fixe de cartes de chaque couleur mais emplacement aléatoire
  - Vérification qu'il n'y a qu'un espion par équipe
  - Vérification qu'il y a assez de joueurs dans chaque équipe
- Déroulement :
  - Formulaire pour envoyer un mot et un nombre aux agents
  - Vérification de l'indice donné par l'espion
  - Sélection d'une carte (unanimement ou non) au clic par les agents
  - Pouvoir '' marquer '' une carte au clic sans la sélectionner mais pour la retenir
  - Vérification de la couleur de la carte sélectionnée
  - Attribution des points en fonction de la couleur de la carte sélectionnée
  - Vérification des conditions de victoire ou de défaite
- Fin :
  - Sauvegarde des stats des joueurs

Tests :

- Tests unitaires
- Tests d'intégration
- Tests système

Timeline : (estimation minimale)

1. Les bases (= modélisation) (3 jours)
2. Génération des éléments du jeu (visuelle et dans le code) (3 jours)
3. Fonctionnalités en jeu (sans contexte ni règles) (14 jours)
4. Implémentation (et vérifications) des règles (14 jours)
5. Mise en ligne du jeu (web) (5 jours)
6. Les tests (tout au long du développement à partir du 3. Fonctionnalités en jeu)

5.1.2 POO-FE

Liste des tâches principales :

Browser **:**

- Liste des serveurs : rejoindre partie, nombre de joueurs, thèmes utilisés
- Filtres et tri (MDP ou non, par thèmes)
- bouton : création de partie (champs : nom de partie, mot de passe)

Lobby **:**

- Joueurs et rôles
- Chat joueurs
- Paramètres partie (nombre de joueurs, nombre d'espions par équipe, nombre de cartes à faire deviner par équipe, thèmes de la partie, langages mots, nombre de cartes dans la partie)
- Boutons : répartition équipe (aléatoire ou non)
- Boutons : chat vocal

Jeu **:**

- Joueurs et rôles
- Choix équipe qui commence
- Chat joueurs
- Cartes de couleurs pour les espions et neutre pour les autres
- Chronomètre
- Historique

- Bouton : recommencer partie
- Interface de saisie espion
- Emplacement pour l'indice et le nombre

Editeur de thème **:**

- Saisie nom du thème et mots contenus à l'intérieur

Timeline (estimation minimale) :

1. Familiarisation avec Unity : (3 jours)
2. Création du jeu : (5 jours)
   - Interface de jeu
   - Création des cartes et changement de couleurs
3. Création lobby : (3 jours)
   - Interface lobby
4. Création du browser : (3 jours)
   - Interface browser
5. Création éditeur de thème : interface éditeur (2 jours)

5.2 BDD : Tâches et timeline

Liste des tâches principales :

Analyse des besoins :

- Définir les besoins avec l'équipe de POO-BE
- Définir les relations entre les données qui seront traitées par la BDD
- Documentation

Création de la BDD :

- Créer les tableaux et implémenter les contraintes
- Mettre en pratique et fournir des APIs aux autres équipes

Tests :

- Tests unitaires
- Tests d'intégration
- Tests système

Timeline (estimation minimale) :

1. Analyse du besoin (8 jours)
2. Création BDD (8 jours)
3. Mise en pratique (4 jours)
4. Optimisation : tout au long du développement
5. Tests : en fonction du progrès du groupe

5.3 RES : Tâches et timeline

5.3.1 RES-SC

Liste des tâches principales :

Mécanisme de communication :

- Distribution des IP : Port pour chaque serveur
- Définition du protocole d'envoi de messages
  - Définitions de toutes les actions de communication possible

        - Entre serveurs
        - Serveur - Client
    - Définition du format de paquet de message

Définition des “mock response” pour permettre les autres équipes à faire des tests et développement

Confirmation des besoins de DB (Avec l'équipe de BDD)

- Définitions de tous les data à stocker dans DB
- Préciser la fréquence de consultation et modification

Serveur :

- Construction du modèle de serveur
    - Les composants de niveau bas dans le modèle RPC
        - Couche de transmission
        - Couche de sérialisation et désérialisation
        - Définition de l’interface de la couche application

Réalisation de serveur de connexion

- Support de l'authentification
- Redirection des requêtes au besoin

. Réalisation de serveur de jeu principal (logic server)

. Réponse de toutes les requêtes dans une partie démarrée

. Maintenance de l’état du jeu

. Réponse aux actions des joueurs

- Réalisation de serveur de “Queuing”
    - Gestion de l’état du salon et l’état du joueur
    - Réponses de toutes les requêtes dans un salon quand le jeu n’est pas démarré

Tests :

- Test d’intégration
- Load balancing
- Test de stabilité
- Optimisation de la stabilité
- Optimisation de performance
- Optimisation de déploiement

Timeline (estimation minimale) :

1. Mécanisme de communication (14 jours)
2. Modèle de serveur + serveur de connexion (7 jours)
3. Serveur principal (7 jours)
4. Serveur de la file d’attente (= Queuing server) (7 jours)
5. Tests et optimisation (14 jours)

5.3.2 RES-SBDD

Liste des tâches principales :

Conception du mécanisme d’interaction :

- Se mettre d’accord avec l’équipe BDD pour choisir les APIs adéquats et les technologies qui sont compatibles avec leurs structures de données.

Confirmation et création du mécanisme

Serveur BDD

- Conception du prototype de serveur BDD et trouver la possibilité de réaliser solution potentielle optimisé par exemple «Read/Write Splitting »
- Réalisation du serveur
- Valider le résultat avec l'équipe BDD

Tests et optimisation

Timeline (estimation minimale) :

1. Mécanisme d'interaction (14 jours)
2. Serveur BDD (21 jours durant tout le développement de BDD)
3. Tests et optimisations (14 jours)

6. GANTT

Pour le GANTT, les tâches ont été réparties en fonction de différentes phases de développement : conception, implémentation, intégration, validation et documentation. La phase d'implémentation, elle-même, est répartie en fonction des équipes de développement. Les équipes travailleront en parallèle afin de réaliser, en premier temps, une version alpha suivie d'une version Beta et enfin la version finale à rendre. Chaque version représentera un jalon.

- La version Alpha sera un prototype primitif du jeu qui consistera à implémenter les règles du jeu implémenté ainsi que les premières versions des interfaces graphiques : lobby, browser et le salon du jeu.
- La version Beta sera plus complétée puisqu'elle inclura la base de données ainsi que les serveurs nécessaires pour le multijoueur.
- Finalement, la version Beta passera par les phases d'intégration et validation ; elle sera optimisée et mise à jour selon les besoins et les tests effectués afin de pouvoir créer la version finale.

Le GANTT sera inclus en version PDF.