

Documentation du projet « Imposteur »

1. Gestion de Projet

1.1 Répartition des Tâches

Membre	Rôle	Tâches principales
Naman Nguyen	Chef de projet / Développeur	Tout

1.3 Prévisions

Phase	Durée estimée	Échéance
Spécifications	3 jours	05/05/2025
Développement (serveur/client)	3 semaines	26/06/2025
Tests & corrections	2 semaines	11/06/2025
Livraison et présentation	-	13/06/2025

1.4 Outils utilisés

- VSCode
- WSL
- Github
- SSH
- IA (ChatGPT, Grok3, Claude...)

2. Installation et démarrage des Applications

2.1 Prérequis

- **CMake** : `sudo apt install cmake`

2.2 Compilation

Pour compiler le serveur :

- `cd server && make`

Pour compiler le client :

- `cd client`
- `mkdir build && cd build`
- `cmake ..`

- make

2.1 Lancement du Serveur

```
./serveur [-p PORT] [-j NB_JOUEURS] [-r NB_TOURS] [-t TIMING_PLAY] [-T TIMING_CHOICE]
```

- **PORT** : port d'écoute (défaut : 5000)
- **NB_JOUEURS** : nombre de joueurs attendus (défaut : 10)
- **NB_TOURS** : nombre total de tous (défaut : 3)
- **TIMING_PLAY** : délai max pour proposer un mot en secondes (défaut : 30)
- **TIMING_CHOICE** : délai max pour désigner l'impateur (défaut : 60)

2.2 Lancement du Client

```
./client -s IP [-p PORT]
```

- **IP** : IP du serveur
- **PORT** : port du serveur (défaut : 5000)

3. Protocole

3.1 Commandes Serveur → Client

Commande	Description
/login	Demande au client de s'identifier
/assign WORD	Donne au joueur le mot secret
/ret ACTION:CODE	Code retour suite à une action
/info ...	Informations serveur (voir variantes ci-dessous)
/play TIMEOUT	Demande au joueur de dire un mot
/choice TIMEOUT	Demande au joueur de désigner un imposteur

Variante /info ...	Description
ID:server_id	Identifiant ou nom du serveur
LOGIN:n/N:login	Annonce d'un joueur connecté (n° sur N)
GAME :r/R:nb_players:timeout_play:timeout_choice	Début d'un tour

WAIT:LOGIN:ACTION	Attente d'une action de LOGIN
SAY:LOGIN:word	Annonce d'un mot lié par un joueur
CHOICE :LOGIN1 :LOGIN2	LOGIN1 accuse LOGIN2
ANSWER:LOGIN1:mot_impoteur:mot_autres	Annonce de l'impoteur et de son mot ainsi que le mot commun
RESULT:LOGIN1:score1:...	Annonce des scores des joueurs
ALERT:message	Message important

3.2 Commandes Client → Serveur

Commande	Description
/login LOGIN	Propose un identifiant
/play WORD	Propose un mot associé
/choice LOGIN	Désigne un joueur comme imposteur
autre	Non reconnu → /ret PROTO :201

3.3 Codes de Retour

Code	Signification
000	OK
101	Login déjà utilisé
102	Pas votre tour
103	Mot déjà proposé
104	C'est votre propre mot !
105	Auto-accusation impossible
106	Login inconnu
107	Login invalide
108	Mot invalide
201	Commande inconnue
202	Commande non attendue

3.4 Contraintes

Élément	Détails
LOGIN	3-16 caractères, pas de « : »
WORD	Max 32 caractères, pas de « : »
PORT	Si < 1024, droits root requis

4. Architecture

4.1 Structure du projet

Imposteur/

```
├── .gitignore
├── client/
│   ├── CMakeLists.txt
│   └── src/
│       └── main.cpp
├── README.md
└── server/
    ├── data/
    │   └── words.csv
    ├── imposteur_server
    ├── include/
    │   ├── color.h
    │   ├── config.h
    │   ├── game.h
    │   ├── player.h
    │   └── utils.h
    ├── Makefile
    └── src/
        ├── game.c
        ├── imposteur_server.c
        ├── player.c
        └── utils.c
```

4.2 Découpage fonctionnel du server

1. Serveur principal (imposteur_server.c)

- Initialise le jeu et les sockets
- Gère les connexions et les messages des joueurs
- Coordonne les phases du jeu (login, jeu, vote, résultats)

2. Module de jeu (game.c/.h)

- Gère l'état de la partie (phases, rounds, imposteur, mots joués)

- Assigne les mots secrets
- Gère les propositions de mots et les votes
- Réinitialise le jeu en fin de partie

3. Module joueur (player.c/.h)

- Gère la liste des joueurs (ajout, suppression, recherche)
- Stocke leurs infos : pseudo, mot secret, score, vote, état prêt

4. Utilitaires (utils.c/.h)

- Diffuse des messages à tous les joueurs
- Journalise les événements serveur
- Parse les commandes reçues
- Sélectionne des mots aléatoires depuis words.csv

5. Fichiers de configuration

- **config.h** : paramètres globaux (ports, délais, tailles)
- **color.h** : couleurs ANSI pour la console

4.3 Découpage fonctionnel du client

1. Logique de jeu (structure et états)

- **GameData** : contient tous les états du jeu côté client (joueurs, mots, scores, phase actuelle, etc.).
- **GAME_STATE** : enum pour représenter les différentes phases du jeu (login, assignation, jeu, vote, résultats...).

2. Connexion réseau

- **connect_to_server()** : établit une connexion TCP avec le serveur.
- **send_message()** : envoie des commandes au serveur (/login, /play, /choice...).
- **handle_server_messages()** : lit les messages du serveur, les interprète et met à jour l'état du jeu.

3. Interface utilisateur (UI avec FTXUI)

- Rendu dynamique de l'écran selon l'état du jeu (splash, login, jeu, vote, résultat...).
- **Composants interactifs** : champs texte et boutons (login_component, play_component, choice_component).
- **render** : construit l'interface complète avec les panels (infos, joueurs, journal d'événements...).

4. Gestion des timers et splash

- Splash screen de démarrage (splash_component) pendant 5 secondes.

- Timer de jeu et de vote avec barre de progression (durée configurable).
- Redessin de l'écran en continu avec animation spinner.

5. Commandes et parsing

- **parse_input()** : découpe les lignes /commande:param1:param2 reçues du serveur.
- Gestion des commandes /login, /assign, /play, /choice, /ret, /info, etc.

4.4 Librairies externes

- FTXUI (pour l'affichage) : [Github](#)