# Fonctionnement générale de l’application

Le but de l’application développée dans le cadre du projet GEN est de recréer le jeu Pictionary. Cette application sera élaborée en vue d’une utilisation sur un réseau local car c’est un jeu par définition multi-joueurs.

Pour rappel, le jeu traditionnel se joue par équipe. A chaque tour, un joueur tente de faire deviner un mot à l’aide d’un dessin si son équipe trouve le mot alors elle gagne un point. Le temps pour effectuer le dessin est limité. De plus, le joueur n’a pas le droit d’écrire le mot à faire deviner. Après un certain nombre de tour, la partie est finie et on compte les points pour désigner la meilleure équipe.

Notre objectif de base est de fournir une application qui doit pouvoir :

* Gérer des parties sur un serveur. Cela comprend une gestion des équipes et donc des joueurs connectés.
* Offrir une interface graphique pour dessiner un mot tiré au hasard.
* Minuter le temps pour le dessin.
* Afficher un chat simple pour afficher les propositions de mots.
* Stocker un dictionnaire qui sera utilisé pour sélectionner un mot.
* Gérer les scores.

Il y aura deux modes de jeux, soit par équipe soit tous contre tous. Commençons par expliquer le premier cas.

Un joueur pourra soit créer une partie, soit en rejoindre une. Le joueur créant une partie pourra spécifier le nombre de tours pour la partie.

Les autres joueurs pourront rejoindre une partie créée en indiquant l’IP du serveur et rejoindre une des deux équipes (bleu ou rouge). Une fois tous les joueurs dans la partie, le créateur de la partie pourra décider de commencer la partie. Ensuite une des équipes sera tirée au sort pour savoir qui va commencer à jouer. Au sein de l’équipe qui commence à jouer, il y a un autre tirage au sort pour désigner qui sera le dessinateur. Le dessinateur recevra un message pour lui avertir de son rôle et le mot qu’il doit dessiner. Le dessinateur aura accès au dessin et non au chat tandis que les « devineurs » pourront écrire dans le chat et voir aussi les propositions de l’équipe. Une fois que le dessinateur commence à dessiner, les autres joueurs de la partie (les deux équipes) vont voir le dessin. L’équipe qui commence pourra utiliser le chat et aura 1 minute pour trouver le mot si elle n’y arrive pas l’autre équipe, qui pendant cette minute à son chat désactivé, pourra elle aussi proposer des solutions, à l’aide du chat qui se sera réactivé pour trouver le mot. La partie se termine une fois que le mot a été trouvé ou que le temps général (pour les deux équipes) qui est de 2’30 est écoulé. Ensuite les équipes vont s’échanger de rôle. La partie s’arrête lorsque le nombre de tour a été atteint.

Dans le deuxième cas, tous les joueurs verront le dessin et tous les joueurs pourront utiliser le chat. Il y aura juste un tirage au sort pour savoir qui va être dessinateur.

Les utilisateurs pourront lors du lancement de l’application soit choisir de se connecter, si ils n’ont pas de compte ils pourront en créer un, soit d’être « anonyme ». Les joueurs connectés l’avantage d’une gestion de score à l’aide d’une base de donnée, tandis que les utilisateurs anonyme n’auront pas la fonctionnalité des scores.

Côté serveur, l’administrateur pourra gérer le dictionnaire (ajouter, modifier, supprimer des mots). Il pourra aussi voir les parties en cours.

L’application sera développée en vue d’une utilisation dans un réseau LAN. Ce qui implique que l’adresse IP du serveur sera connue par les utilisateurs qui se connecteront à l’application. On pourrait imaginer un mécanisme où l’application cliente se connecte sur un serveur web par une URL et qui en suite serait redirigée vers un serveur mais cela ne correspond pas à notre environnement de développement.

Certaines règles de jeu pourraient être facilement ignorées.

# Communication client-serveur

Le rôle du serveur est de gérer les connexions des joueurs et de gérer les parties en cours. Le rôle de l’application client est de fournir une interface pour que l’utilisateur puisse dessiner et utiliser le chat. Le serveur retransmet les informations aux applications clientes.

Lancement du jeu : Les joueurs se connectent au serveur. Une fois connectées, ils ont le choix de créer une partie ou d’en rejoindre une déjà créée.

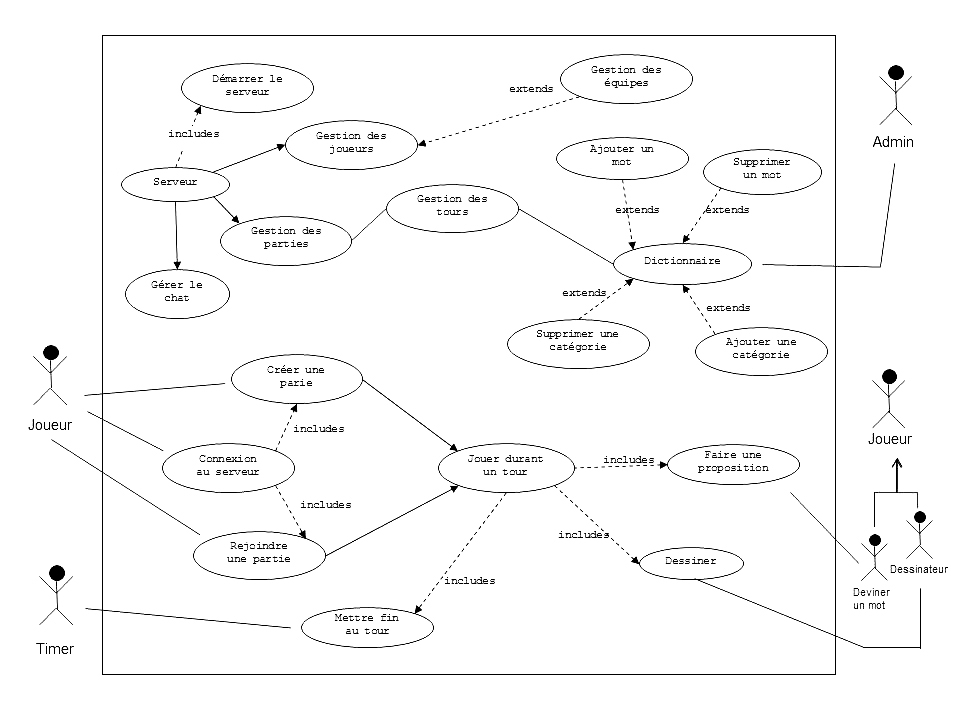
Fin du jeu : Lorsque le nombre de tours est écoulé, la partie prend fin. L’utilisateur est ramené à l’écran d’accueil qui permet de créer une partie ou d’en rejoindre une.

Le serveur aura un rôle de dieu, il sera tout puissant et c’est lui qui enverra les signaux pour contrôler les joueurs (début et fin de partie). Sauf quand un client trouvera le mot c’est lui qui va envoyer un signal au serveur pour lui dire qu’il peut envoyer un signal de fin aux autres.

# Cas d’utilisation

## Diagramme de contexte

Voici le fonctionnement général de l’application :



## Description des acteurs

### Acteur principal : Joueur

* Le joueur de l’application peut démarrer une partie ou rejoindre une partie. Pour cela, il doit obligatoirement être connecté au serveur.
* Le jouer peut avoir deux rôles : Soit dessiner un mot tiré du dictionnaire soit deviné le mot tiré du dictionnaire et faire des propositions sur le chat.

### Acteur secondaire : Timer, Administrateur

* Le Timer met fin au tour d’une partie. Une partie est composée de plusieurs tours.
* L’Administrateur peut gérer les mots proposés au joueur et les classer par catégorie.

## Scénarios principales de succès

### Dictionnaire

1. Ajouter des mots au dictionnaire
2. Supprimer des mots au dictionnaire
3. Ajouter une catégorie
4. Supprimer une catégorie

### Connexion au serveur

1. L’utilisateur se connecte au serveur via son adresse IP.
2. L’utilisateur sélectionne un pseudo.
3. Une fois connecté, il arrive sur un écran pour la gestion des parties.

### Créer une partie

1. Une fois l’utilisateur connecté au serveur, il peut créer une partie.
2. L’utilisateur sélectionne son mode de jeu (par équipe / tous contre tous)
3. L’utilisateur est devient administrateur de la partie.
4. Le serveur attend d’autres connexions pour permettre le lancement de la partie.
5. L’utilisateur lance la partie une fois les autres joueurs connectés.

### Rejoindre une partie

1. Une fois l’utilisateur connecté au serveur, la liste des parties qui peuvent être rejointes sont affichées.
2. L’utilisateur rejoint une partie.
3. Si l’utilisateur a sélectionné une partie en équipe il choisit son équipe.
4. La partie commencera lorsque l’utilisateur qui a créé la partie la lancera.

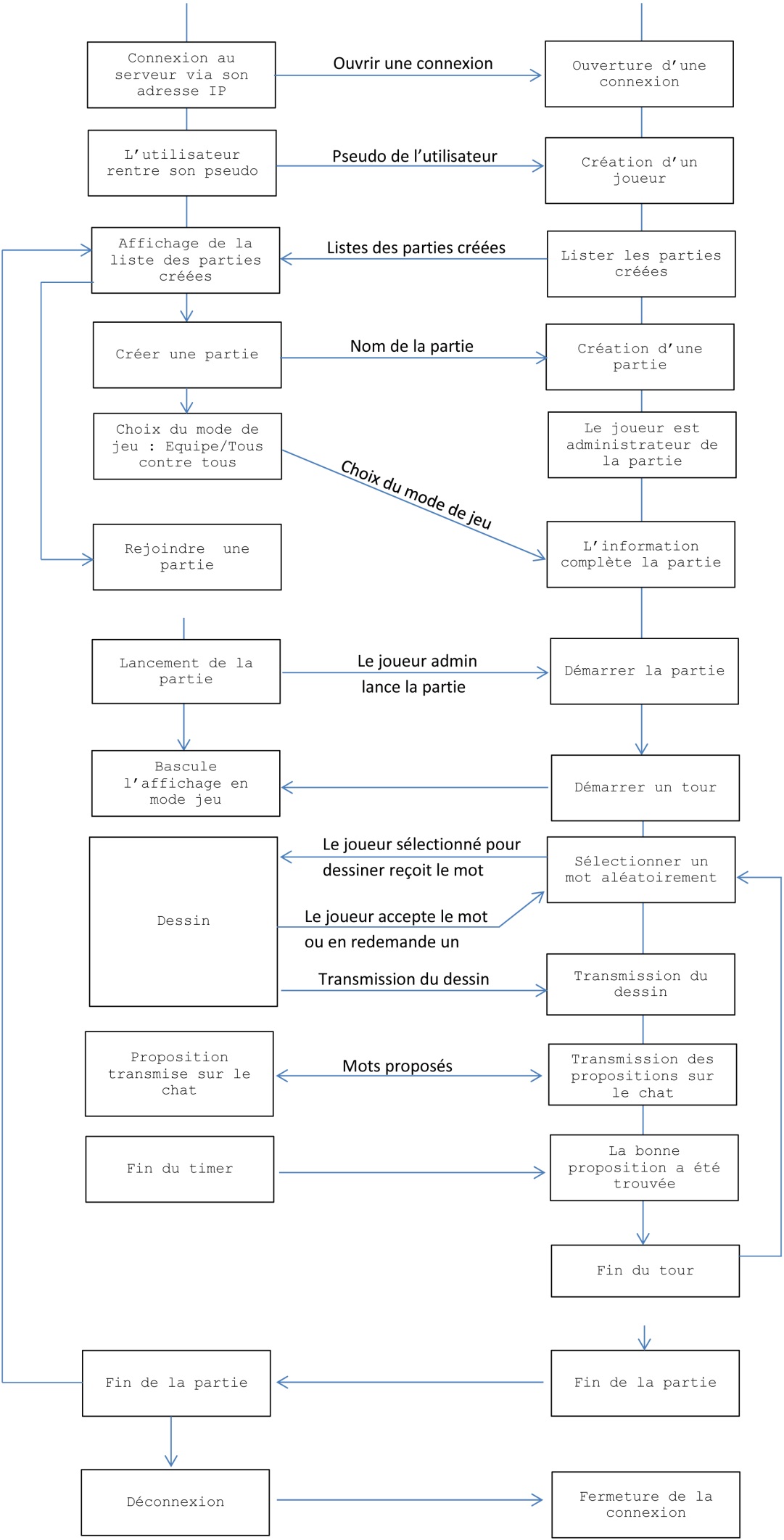
### Jouer durant un tour

1. Une fois la partie lancée, le premier tour commence.
2. L’utilisateur se voit attribuer un mode de jeu pour ce tour :
3. Mode dessinateur :
   1. S’il est dessinateur, le joueur choisit une catégorie.
   2. Il reçoit un mot à dessiner.
   3. S’il ne connait pas ce mot, il peut en demander 2 autres au maximum.
4. Mode joueur :
   1. Si l’utilisateur ne dessine pas il doit deviner le mot qui a été attribué à un autre joueur.
   2. Il peut faire des propositions dans le chat.
5. Le tour se termine :
   1. Si un joueur a trouvé le bon mot.
   2. Si le timer indique que le temps est écoulé.
6. Il gagne des points en fonction de son mode de jeux
7. Les opérations ci-dessus se répètent pendant un nombre de tour prédéterminé.

# Protocole de communication entre le client-serveur

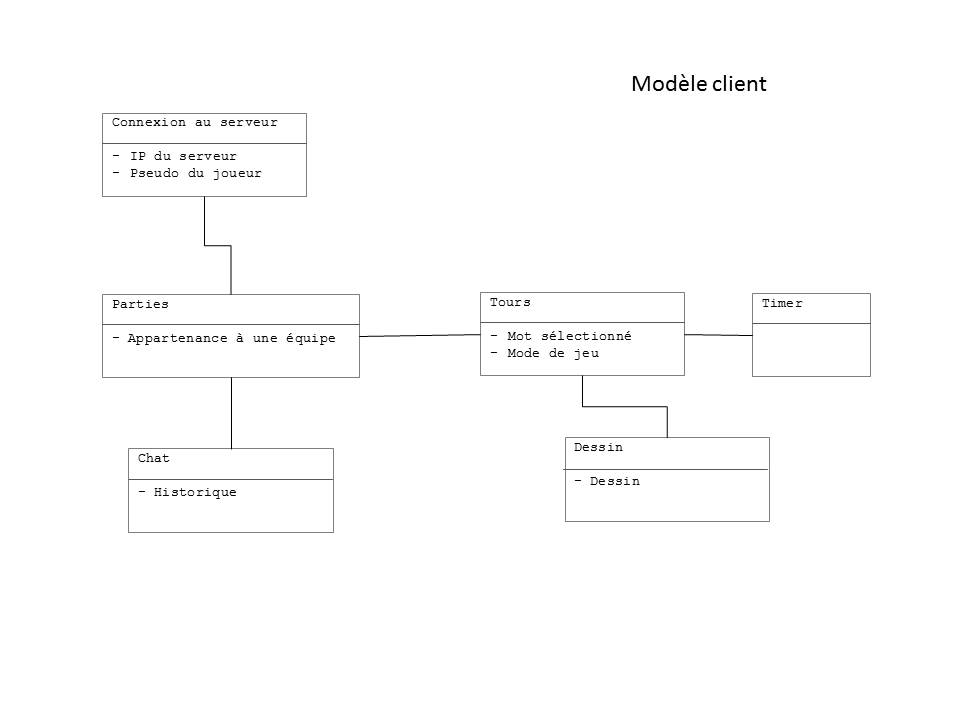
CLIENT

SERVEUR



# Modèle de domaine





# Base de donnée et conception du dictionnaire

Pour le fonctionnement de l’application, il faut tirer un mot aléatoirement issu du dictionnaire. Les mots tirés au hasard sont classés par catégorie et le joueur peut choisir la catégorie pour son tour.

La structure de donnée à mettre en place n’est pas particulièrement complexe à mettre en place. Les données seront donc stockées dans un fichier XML.

Dictionnaire

Listes de catégories

Liste de mots

# Groupe de développement

Il y a 4 personnes qui composent le groupe de développement.

|  |  |  |
| --- | --- | --- |
| **Rôle standard** | **Responsabilités** | **Responsable** |
| Représentants des utilisateurs | * Collecte des besoins pour le projet. * Spécifications des tests de fonctionnalités * Explication des aspects métiers | [Melly](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Calixte |
| Chef de projet | * Planification * Coordinations avec les utilisateurs | Fröhlich Magali |
| Analyste | * Spécifications * Collecte des demandes de changement | Righitto [Simone](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) |
| Architecte,  concepteur en chef | * Conception de l’architecture du produit | [Saam Frédéric](https://webmail.heig-vd.ch/owa/?ae=Item&t=IPM.Note&a=New),  Fröhlich Magali |
| Programmeur | * Participe à la conception du produit * Ecrit les tests unitaires * Codage | [Righitto](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Simone,  [Saam Frédéric](https://webmail.heig-vd.ch/owa/?ae=Item&t=IPM.Note&a=New),  Fröhlich Magali,  [Melly](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Calixte |
| Responsable des tests | * Participe à l’intégration continue des composants * Ecrit les tests fonctionnels * Met en place l’architecture permettant de lancer régulièrement les tests fonctionnels | [Righitto](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Simone |
| Responsable de la configuration | * Gestion de la base des artefacts du projet * Gestion des releases * Allocation des droits * Responsable de la configuration (logicielle & matérielle) * Intégration des changements | [Saam Frédéric](https://webmail.heig-vd.ch/owa/?ae=Item&t=IPM.Note&a=New) |

# Itération

|  |  |
| --- | --- |
| **Semaine** | **Objectif et déroulement** |
| 28 avril 2014 | Rendu du rapport intermédiaire |
| 05 mai 2014 | Objectif : Connexion au serveur, Dictionnaire  [Righitto](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Simone : Analyse et création du fichier XML pour la gestion des mots qui composent le dictionnaire.  [Saam Frédéric](https://webmail.heig-vd.ch/owa/?ae=Item&t=IPM.Note&a=New) : Conception du serveur et de l’application cliente.  Fröhlich Magali : Documentation et programmation des classes de base pour le serveur.  [Melly](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Calixte : Programmation des classes de base pour l’application cliente. |
| 12 mai 2014 | Objectif : Gestions des joueurs, Gestion des équipes  [**Righitto**](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) **Simone** : Programmation des classes pour la connexion des joueurs du côté serveur.  Saam Frédéric : Conception du modèle observable-observé pour préparer les interfaces graphiques.  Fröhlich Magali : Programmation des classes pour la création des équipes du côté serveur.  [Melly](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Calixte : Programmation des interfaces graphiques pour le serveur et pour le client. |
| 19 mai 2014 | Objectif : Création d’une partie, Rejoindre une partie, Dictionnaire  [Righitto](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Simone : Conception du dictionnaire et réalisation des méthodes pour ajouter/supprimer des mots, ajouter/supprimer des catégories.  [Saam Frédéric](https://webmail.heig-vd.ch/owa/?ae=Item&t=IPM.Note&a=New) : Conception des classes pour la gestion des parties.  Fröhlich Magali : Programmation des classes pour la gestion des parties : Création des parties.  [Melly](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Calixte : Programmation des classes pour la gestion des parties : Rejoindre une partie. |
| 26 mai 2014 | Objectif : Dessiner, Chat  [Righitto](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Simone : Programmation de l’interface graphique générale.  [Saam Frédéric](https://webmail.heig-vd.ch/owa/?ae=Item&t=IPM.Note&a=New) : Conception des classes pour le dessin et le chat.  Fröhlich Magali : Programmation de l’interface graphique pour dessiner.  [Melly](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Calixte : Programmation de l’interface graphique pour écrire dans le chat. |
| 02 juin 2014 | Objectif : Dessiner, Chat  [Righitto](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Simone : Programmation des échanges pour activer les différents modes  (mode dessinateur ou joueur).  [Saam Frédéric](https://webmail.heig-vd.ch/owa/?ae=Item&t=IPM.Note&a=New) : Conception et programmation du Timer et de la gestion des scores  Fröhlich Magali : Programmation des échanges pour afficher le dessin  [Melly](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Calixte : Programmation des échanges pour afficher le chat |

|  |  |
| --- | --- |
| 09 juin 2014 | Objectif : Gérer les tours  [Righitto](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Simone : Conception des algorithmes pour la gestion des mots et des catégories.  [Saam Frédéric](https://webmail.heig-vd.ch/owa/?ae=Item&t=IPM.Note&a=New) :  Fröhlich Magali :  [Melly](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Calixte :   Test générale de l’application : Le but est de vérifier si l’on peut effectuer toutes les actions que l’on pourrait faire avec un vrai jeu. |
| 16 juin 2014 | Objectif : Test final de l’application et documentation  [Righitto](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Simone :  [Saam Frédéric](https://webmail.heig-vd.ch/owa/?ae=Item&t=IPM.Note&a=New) :  Fröhlich Magali :  [Melly](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Calixte : |
| 23 juin 2014 | Présentation et rendu final |