**HEIG-VD**

**GEN**

16

**HEIG- VD**

Projet Génie Logiciel :

Application de chat client/serveur

Rapport intermédiaire

Amel Dussier, Batien Clément, Antoine Drabble et Guillaume Serneels

Table des Matières

Fonctionnement général 3

Partage des résponsabilités clients/serveur 3

Cas d’utilisation 3

Diagramme général de contexte 5

Déscription des acteurs 6

Acteurs principaux 6

Acteurs secondaires 6

Scénario principal pour chaque cas d’utilisation 6

Scénario principal (exemple: démarrage d’une conversation privée par un utilisateur A avec un utilisateur B) 6

Scénario d’échec (exemple problème de connection internet): 6

Protocole d’échange client/serveur 7

Ebauche du modèle de domaine (découpage MVC) 7

Modèle serveur 7

Modèle client 7

Base de données 7

Objectif 7

Modèle conceptuel 7

Ebauche des interfaces utilisateur 8

Fenêtre de connection 8

Fenêtre d’acceuil 8

Fenêtre de chat 9

Fenêtre d’édition de groupe 9

Gestion du projet 9

Role des participants 9

Plan d’itérations 10

Glossaire 10

# Fonctionnement général

Nous allons réaliser une application de chat client-serveur ressemblant à Whatsapp et Telegram. La première fonction sera la création de compte, ou simplement la connection à un compte existant (en fournissant un nom d’utilisateur et un mot de passe). Il y aura ensuite trois types de chat possible :

* Un chat privé 1 à 1 qui permettra de communiquer avec ces contacts. La première version permettra de communiquer par message texte puis si le temps le permet, nous étudierons la possibilité d’implémenter la communication audio et les discussions chiffrée de bout en bout avec un système utilisant des clés publiques/privées. Les contacts pourront être ajoutés en les recherchant par leur nom d’utilisateur.
* Un chat privé de groupe. Un utilisateur pourra créer un groupe avec plusieurs de ses contacts et nommer le groupe. Toutes les personnes ajoutées pourront envoyer des messages dans ce groupe qui seront lisible par tous les membres. Le créateur du groupe en sera l’administrateur et pourra y ajouter/supprimer des membres.

.

* Finalement, il y aura une fonctionnalité de report de messages qui permettra aux utilisateurs de reporter un message abusif. Tous les reports seront envoyés à l’administrateur de l’application et à l’administrateur du groupe si ce n’est pas une discussion 1 à 1. Les reports envoyés à l’administrateur seront stockés dans une base de données SQL et seront accessible via une interface web. Un utilisateur pourra en bloquer un autre afin de ne plus recevoir de messages de sa part. L’application serveur sera codée en Scala, l’application client sera sur Android et les informations seront stockées à l’aide d’une base de données SQL.

# Partage des résponsabilités clients/serveur

Le serveur s’occupe de gérer l’ensemble des opérations, le client n’est qu’une vue des données du système. Il affiche les informations que l’utilisateur demande et transmet les actions qu’il souhaite effectuer. Le serveur s’assure qu’il en aie l’autorisation.

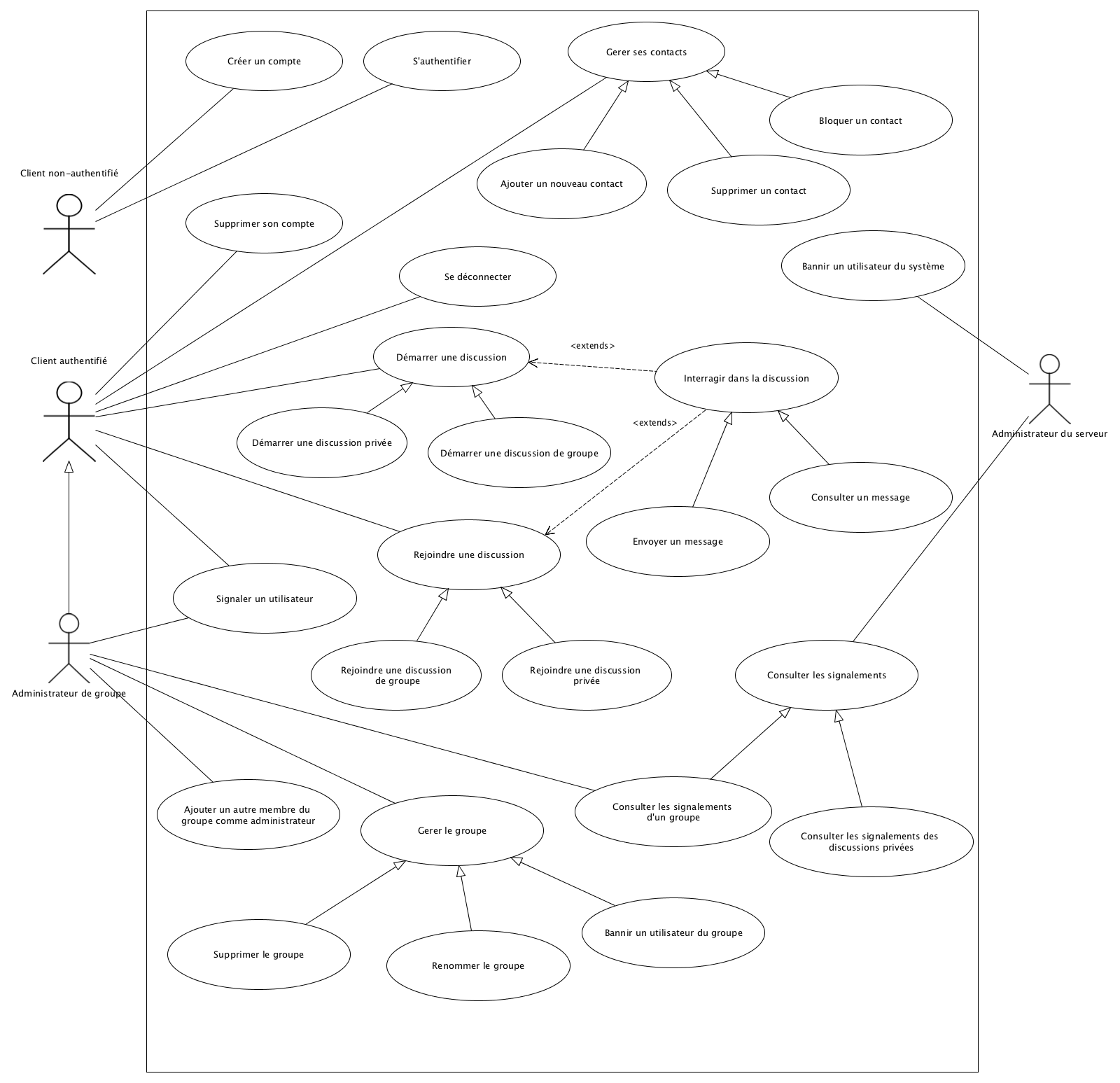
Le serveur fonctionne en permanence. Le client peut à tout moment s’y connecter en démarrant l’application et en fournissant ses informations de connections (nom d’utilisateur et mot de passe).

Si l’application n’est pas lancée, l’utilisateur est considéré comme déconnecté. Après la première connection, le client maintient un cookie (token) de session lui permettant de se reconnecter rapidement

# Cas d’utilisation

* Création de compte
* Connection à l’application
* Rejoindre une discussion publique
* Créer un groupe de discussion public
* Créer une discussion privée
* Supprimer un utilisateur d’un groupe de discussion public
* Consulter les reports d’un groupe publique
* Consulter les reports d’un groupe privé
* Quitter un groupe
  + Quitter un groupe publique
  + Quitter un groupe privé
* Gérer ses contacts
  + Ajouter un nouveau contact
  + Supprimer un contact
  + Bloquer un contact
* Ajouter un utilisateur dans un groupe
* Ajouter un nouveau contact
* Supprimer un contact
* Bloquer un contact
* Reporter un autre utilisateur
* Supprimer son compte
* Envoyer un message
  + Dans un contact
  + Dans un groupe privé
  + Dans un groupe publique
* Consulter les messages
  + Message 1 à 1
  + Message d’un groupe publique
  + Message d’un groupe privé
* Se déconnecter de l’application
* Charger des messages de l’historique

## Diagramme général de contexte



## Déscription des acteurs

### Acteurs principaux

* Client non-authentifé  
  Il devra soit créer un compte soit s’authentifier pour se transformer en acteur authentifié
* Client authentifié  
  C’est l’acteur qui représente les utilisateurs qui possèdent un compte et qui sont connecter
* Administrateur de groupe  
  Il est notifié des reports des discussions de son groupe et peut gérer les membres du groupe. Le rôle d’administrateur de groupe étend son rôle de client authentifié.

### Acteurs secondaires

* Administrateur du système  
  Il est notifié des reports de groupes et de discussions privées et peut bannir des utilisateurs de l’application. Cet acteur n’a pas de rôle de client.

## Scénario principal pour chaque cas d’utilisation

### Scénario principal (exemple: démarrage d’une conversation privée par un utilisateur A avec un utilisateur B)

1. L’utilisateur A ouvre l’application
2. L’utilisateur A se connecte à l’application
3. Le serveur valide la connection
4. L’utilisateur A choisi l’utilisateur B avec lequel il veut discuter et ouvre une conversation
5. L’utilisateur A envoie son/ses messages, l’utilisateur B sera notifié de la nouvelle discussion privée et aura la possibilité de lire et répondre aux messages de l’utilisateur A la prochaine fois qu’il se connectera à l’application. Si ils sont connectés en même temps, ils ont la possibilité d’échanger des messages en temps réel.
6. L’utilisateur A quitte la conversation
7. L’utilisateur A se déconnecte de l’application
8. Le serveur valide la déconnection

### Scénario d’échec (exemple problème de connection internet):

2a. L’appareil n’a pas de connection internet

1. Le Système signale l’erreur et rejette la saisie

2b. Le serveur n’est pas disponible

1. Le système signale l’erreur et rejette la saisie

# Protocole d’échange client/serveur

Le protocole de communication client serveur devra permettre les commandes suivantes :

Client -> Serveur

* Demande de création de compte
* Validation/Refus de création de compte
* Demande de tentative de connection
* Validation/Refus de tentative de connection
* Demande d’ajout d’un contact
* Validation/Refus d’ajout de contact (par exemple s’il est bloqué ou s’il y a une erreur) (Ajout d’un lien dans les deux sens entre les deux contacts, création de discussions vide)
* Demande de création de groupe
* Validation/Refus de création de groupe
* Envoi d’un message à un contact/groupe
* Validation/Refus d’envoi du message
* Demande de déconnection
* Validation/Refus de déconnection

Serveur -> Client

* Envoi d’un message au client
* Acceptation/Refus de réception
* Création d’un groupe dont le client est membre
* Acceptation/Refus de création

# Ebauche du modèle de domaine (découpage MVC)

## Modèle serveur

## Modèle client

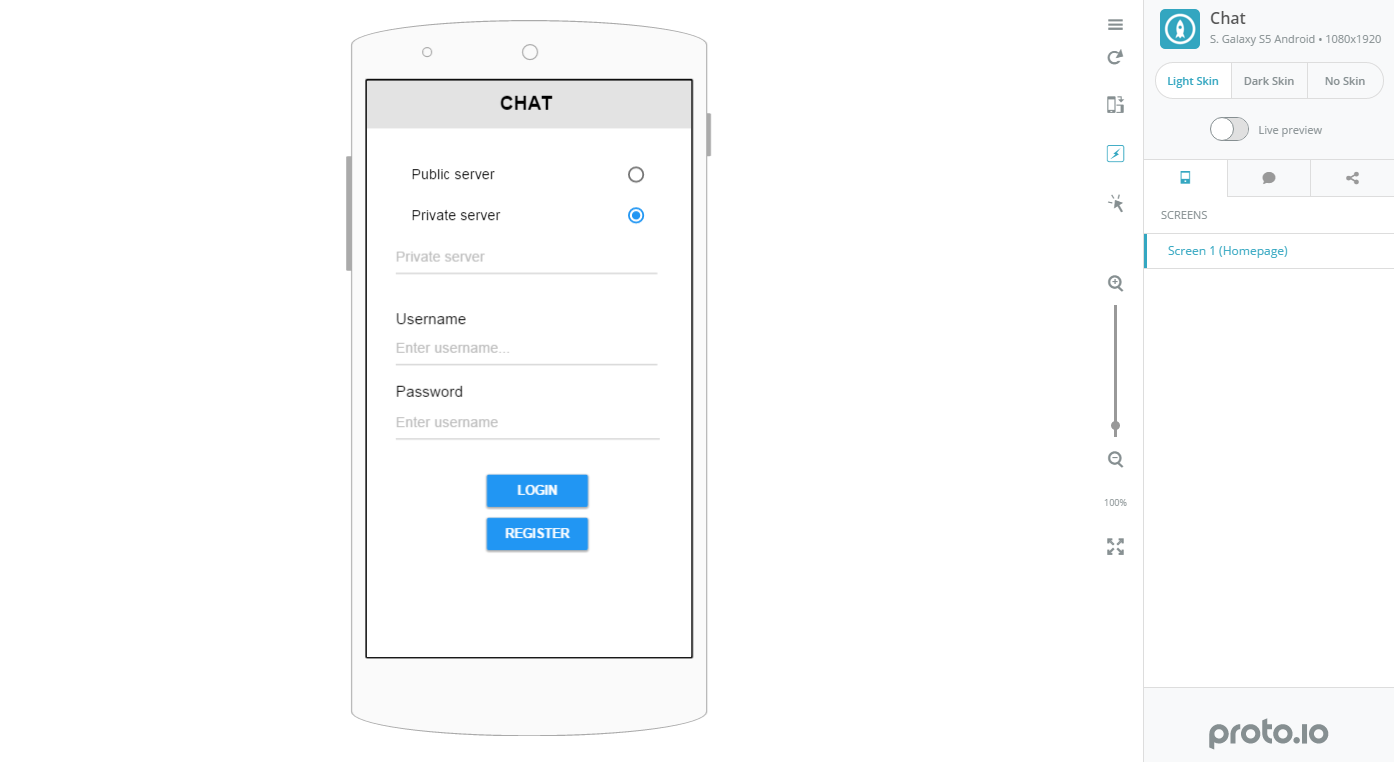
# Base de données

## Objectif

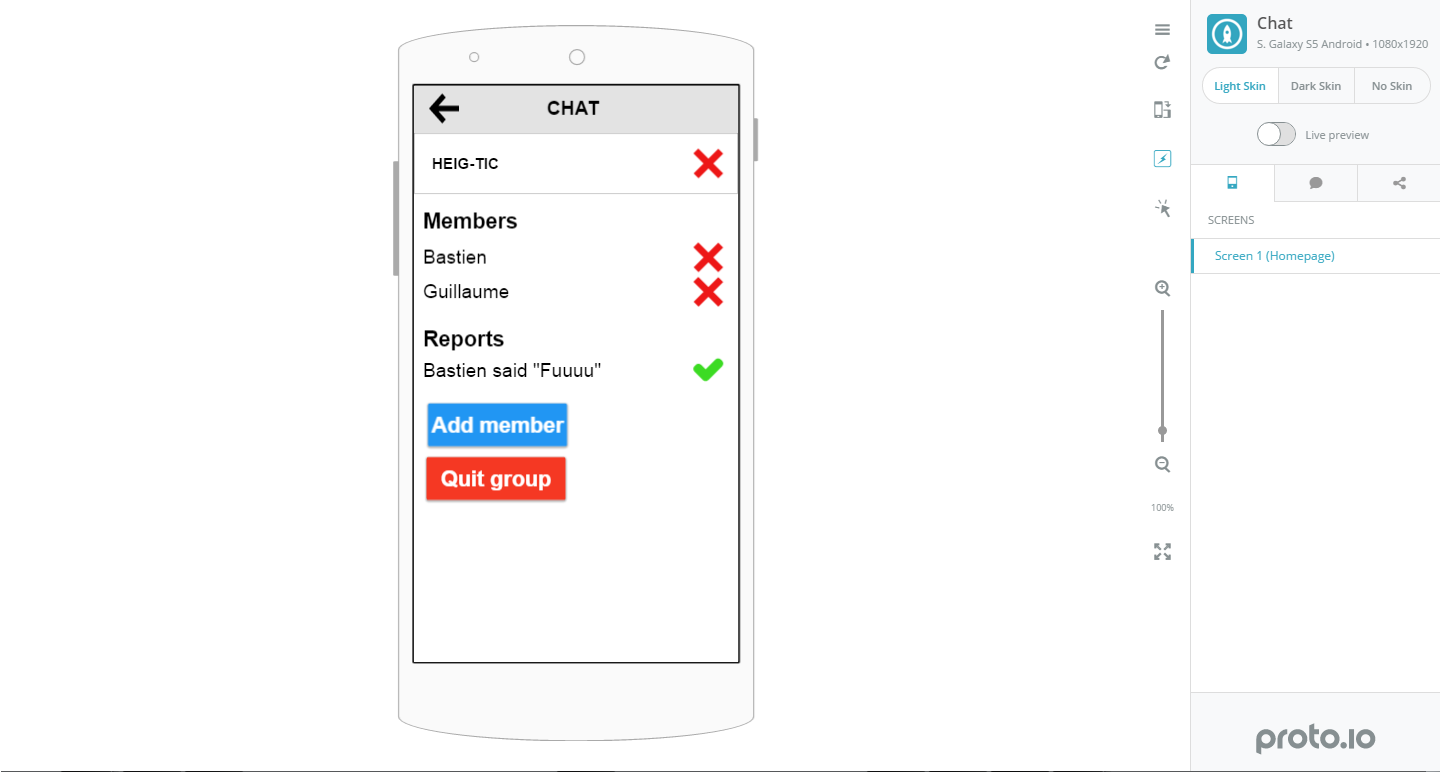
## Modèle conceptuel

# Ebauche des interfaces utilisateur

## Fenêtre de connection



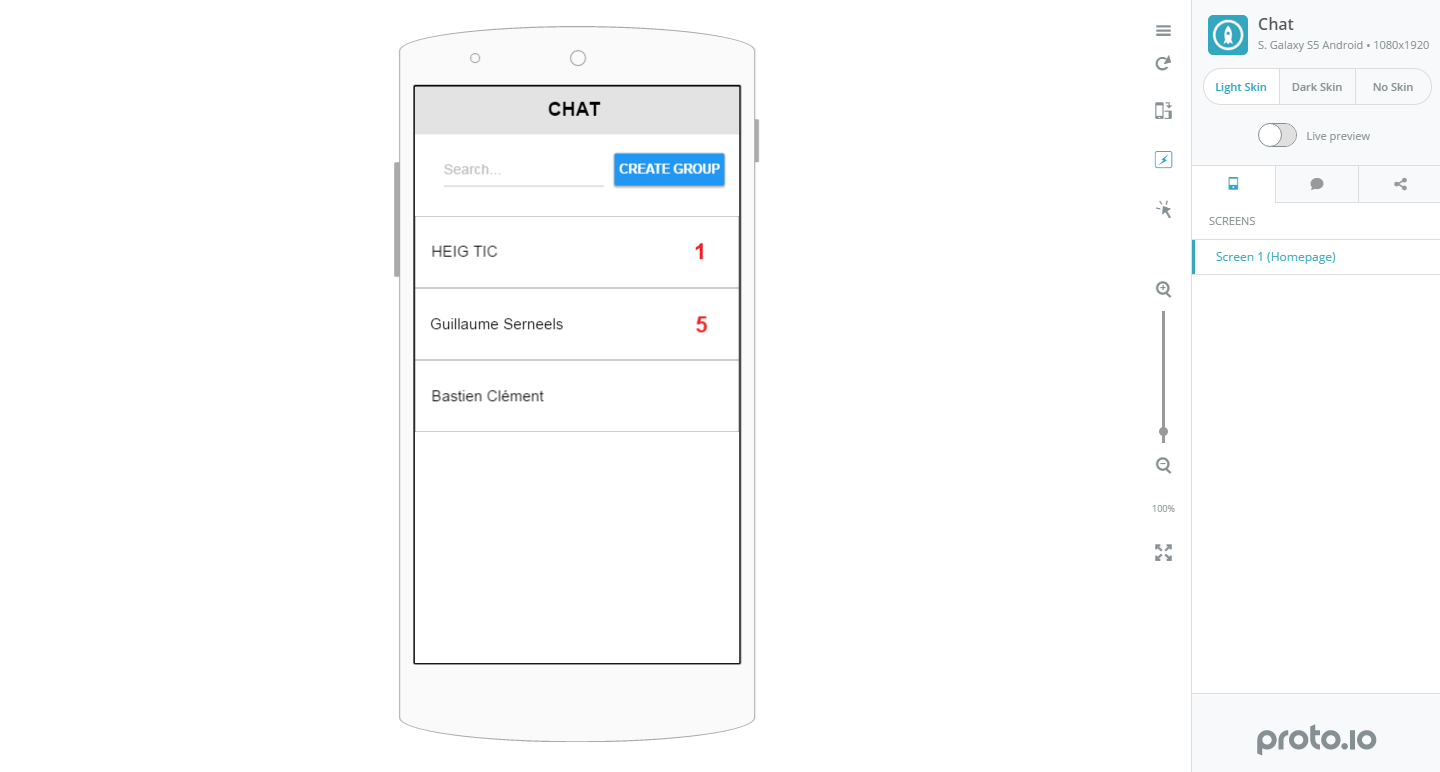
## Fenêtre d’acceuil



## Fenêtre de chat



## Fenêtre d’édition de groupe



# Gestion du projet

## Role des participants

- Représentant des utilisateurs : Client

- Chef de projet : Project Manager

- Analyste

- Software architect

- Programmeur

- Responsable des tests

- Responsable de la configuration

## Plan d’itérations

Pour chaque itération :

- Objectif

- Durée

- Qui fait quoi

- Effort consacré en heures

# Glossaire

**Discussion privée** : Une discussion privée (1 à 1)

**Discussion de groupe** : Une discussion de groupe privée (3 et +)

A VOIR (ajouter quelque part)

Administrateur de groupe quitte le groupe, est bannis par l’admin système, ou supprime son compte, il faut supprimer les groupes