

Étape 2: Proposition

App de Chat – Jazzer (Jaser)

1. Description générale

La principale utilisation de Jazzer sera l'envoi de messages gratuits entre les utilisateurs d'app. En effet, le téléphone mobile besoin être connecté à Internet, via un forfait ou une connexion Wi-Fi. Alors, l'app pourrai l'envoi de l'équivalent d'un SMS gratuit et illimité. Chaque utilisateur doit installer l'application et être connecté à l'Internet pour pouvoir l'utiliser.

2. Les technologies utilisées


































- Java
- IDE Android Studio
- Firebase : Base de données NoSQL – Realtime Database

2.1 Firebase :

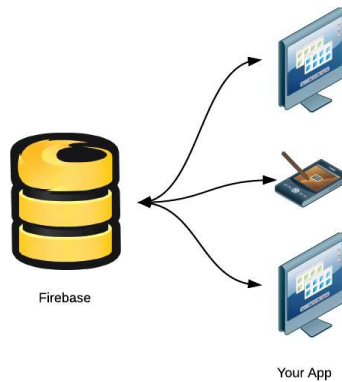
Firebase est une API puissante qui peut être utilisée pour stocker et faire la synchronisation des données en temps réel en utilisant uniquement le code côté client, idéale pour les applications Web et Mobile. Il vous permet de créer des applications sans avoir à gérer des serveurs ou à écrire du code côté serveur. Firebase a été créé en 2009 pour aider les développeurs d'applications, les concepteurs de sites Web et les développeurs mobiles se soucient moins de l'architecture de l'application, authentification et hébergement pour mettre en valeur le produit final et générer une application de qualité rapidement. Les types des services offerts gratuitement aux utilisateurs enregistrés, il y a par exemple un hébergement gratuit de contenu statique (par ex. Pages HTML et fichiers JS), système d'authentification (anonyme, par e-mail ou par réseaux (Facebook, Google+, Twitter, GitHub)) et le stockage de données en format JSON dans une base de données non relationnelle.



Develop & test your app

 Realtime Database iOS    	 Cloud Firestore iOS  
 Authentication iOS    	 Cloud Storage iOS    
 Test Lab 	 Performance Monitoring iOS 
 Crashlytics iOS 	 Crash Reporting iOS 
 Cloud Functions iOS    	 Hosting 

Firebase fournit des bibliothèques clients qui permettent l'intégration avec les applications Android, iOS, JavaScript, Java, Objective-C, swift et Node.js. La base de données est également accessible via une API REST et des liens vers divers frameworks JavaScript, tels que AngularJS, React, Ember.js et Backbone.js. L'API REST utilise le protocole Server-Sent Events, qui est une API permettant de créer des connexions HTTP pour recevoir des notifications push d'un serveur. Les développeurs qui utilisent la base de données en temps réel peuvent protéger leurs données en utilisant les règles de sécurité appliquées par le serveur. (Server-sent events (SSE) sont une technologie dans laquelle un navigateur reçoit des mises à jour automatiques d'un serveur via une connexion HTTP. L'API EventSource des événements envoyés par le serveur est normalisée dans le cadre de HTML5 par le W3C.)



2.1.1 Utilisation:

Pour utiliser Firebase, il faut d'abord créer un compte dans Google Gmail, à travers lequel le service sera accessible. Ensuite, il faut accéder au site web de Firebase : (<https://firebase.google.com>). Le développeur de l'application a la possibilité d'ajouter la base de données aux applications iOS, Android et web. Lorsque vous cliquez sur l'une des options, Firebase fournit le code qui devrait être ajouté à l'application, et en les ajoutant, l'intégration est faite.

2.1.2 Base de données RealTime Database:

Toutes les données de la base de données Firebase – Realtime Database sont stockées en tant qu'objets JSON. Considérez la base de données comme une arborescence JSON hébergée dans le cloud. Contrairement à une base de données SQL, il n'y a pas de tables ou d'enregistrements. Lorsque vous ajoutez des données à l'arborescence JSON, elles deviennent un nœud dans la structure JSON avec une clé associée. Vous pouvez fournir vos propres clés, telles que les codes utilisateur et les noms sémantiques, ou les générer en utilisant push ().

En utilisant le service de base de données en temps réel proposé par Firebase, le développeur du projet doit comprendre un peu plus les bases de données non relationnelles et la structuration des données orientées document.

Le système en temps réel proposé par Firebase fonctionne à partir d'une série de *listeners* par des parties spécifiques d'une collection ou d'un document dans la base de données. Il est possible dans ce modèle de définir que l'application "souhaite être informée" lorsqu'un événement spécifique se produit dans le répertoire de données, comme l'ajout ou la suppression d'une donnée, la modification du contenu du document, etc.

Différent de la façon dont les requêtes fonctionnent dans les bases de données relationnelles, où l'utilisateur accède aux données d'une table, Firebase propose dans les méthodes de l'API d'écouter une partie du document JSON en cours de stockage. Le système de requête requiert une référence à une collection de données, un type d'événement à écouter et un *callback* qui sera appelé chaque fois que Firebase déclenche un événement du type en question.

Les règles Firebase Realtime Database déterminent qui a accès en lecture et en écriture à la base de données, comment les données sont structurées et quels index sont définis. Ces règles résident sur les serveurs Firebase et sont toujours appliquées automatiquement. Chaque demande de lecture et d'écriture ne sera complétée que si les règles le permettent. Par défaut, les règles sont définies pour un accès complet en lecture et en écriture à la base de données uniquement aux utilisateurs authentifiés. La syntaxe des règles de base de données Firebase Realtime Database est similaire à celle de Javascript et a quatre types:

.read Décrit si et quand les données peuvent être lues par les utilisateurs.

.write Décrit si et quand les données peuvent être écrites.

.validate Définit la mise en forme correcte de la valeur, le type de données et si la valeur a des attributs enfants.

.indexOn Spécifie un enfant comme index afin que le tri et l'interrogation soient possibles.

Informations supplémentaires

Langage d'implémentation Firebase : inconnu

Langages de programmation supportés : Java, JavaScript, Objective-C

Côté serveur scripts : Fonctionnalité limitée par le serveur avec l'utilisation de 'règles'

Déclencheurs : Callbacks sont déclenchés lorsque les données changent

Firebase: Primary database model: Document store

Document store, également appelés systèmes de base de données orientés document, se caractérisent par une organisation des données sans schéma. Cela signifie que les enregistrements n'ont pas besoin d'avoir une structure uniforme, c'est-à-dire que différents enregistrements peuvent avoir des colonnes différentes. Les types de valeurs des colonnes individuelles peuvent être différents pour chaque enregistrement. Les colonnes peuvent avoir plusieurs valeurs (tableaux). Les enregistrements peuvent avoir une structure imbriquée. *Document stores* utilisent souvent des notations internes, qui peuvent être traitées directement dans les applications, principalement JSON. Bien entendu, les documents JSON peuvent également être stockés sous forme de texte pur (dans key-value stores) ou des systèmes de bases de données relationnelles. Cela nécessiterait cependant un traitement côté client des structures.

Font :

<https://www.firebase.com/docs/web/api/query/on.html>

<https://db-engines.com/en/system/Firebase+Realtime+Database%3BGoogle+Cloud+Bigtable%3BOracle>

<https://www.firebase.com/docs/rest/guide/>

<https://firebase.google.com/docs/database/security>

<https://www.firebase.com/docs/android/>

<https://www.firebase.com/docs/android/guide/understanding-data.html>

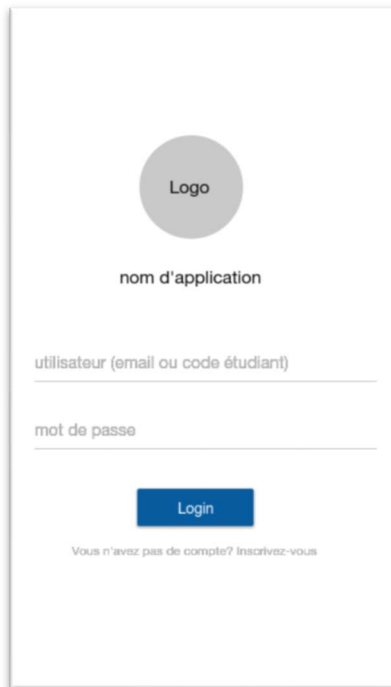
3. Description fonctionnelle

- Créer une Activité de Inscription d'utilisateurs (Une page d'activité où l'utilisateur va créer le compte pour utiliser l'application).
- Créer une interface de connexion
- Ajouter et répertorier les contacts
- Démarrer une conversation
- Lister les conversations et les messages (Liste de conversation: Liste des conversations entre utilisateurs, liste des messages: Historique des messages échangés entre deux utilisateurs)

Scénarios nominaux

Login

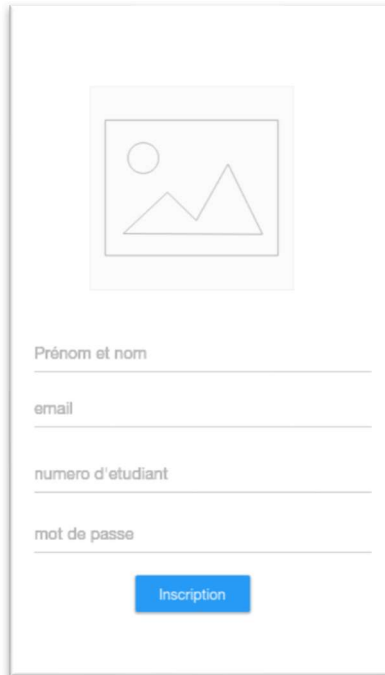
1. L'utilisateur démarre l'application sur son smartphone
2. L'utilisateur tape ses informations de login (email / code étudiant et mot de passe)
3. Il est redirigé vers la page de principal.
4. S'il y a des conversations, la liste des conversations sera affichée



A vertical rectangular mockup of a login screen. At the top center is a gray circle containing the word "Logo". Below it is the text "nom d'application". Further down are two input fields: the first is labeled "utilisateur (email ou code étudiant)" and the second is labeled "mot de passe". Below these fields is a blue rectangular button with the text "Login". At the very bottom, centered, is the text "Vous n'avez pas de compte? Inscrivez-vous".

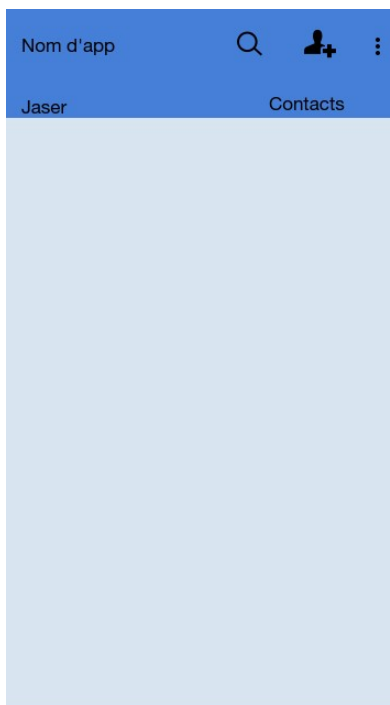
Inscription

1. Un utilisateur clique sur le lien « Inscription ».
2. Il est redirigé vers l'activité d'inscription.
3. Il entre ces informations (Nom d'utilisateur, mot de passe, adresse courriel, etc.) et soumet son inscription.
4. Un Toast message lui confirme son inscription.



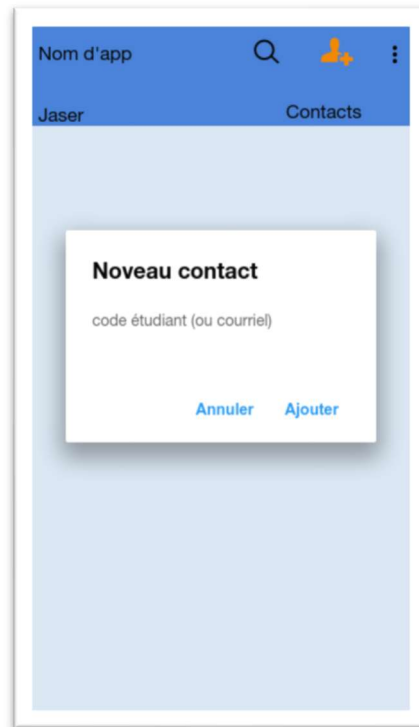
A registration form interface with a light gray background. At the top, there is a square placeholder for a profile picture, containing a simple line-art icon of a mountain and a sun. Below this, there are four text input fields, each with a label above it: "Prénom et nom", "email", "numero d'etudiant", and "mot de passe". The "mot de passe" field has a small eye icon to its right. At the bottom of the form, there is a blue button with the text "Inscription" in white.

5. L'Utilisateur est redirigé vers l'activité d'accueil

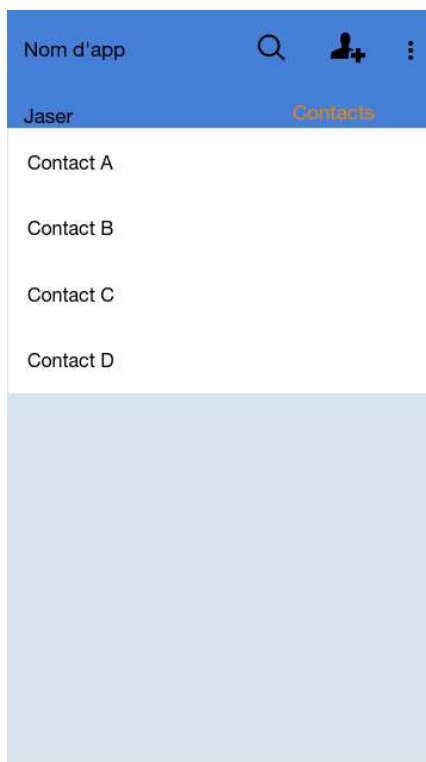


Ajouter un contact

1. Un utilisateur clique sur le Icon d'ajouter un contact
2. S'affichera une boîte de dialogue que demandera l'information nécessaire.
3. Il entre les informations (code étudiant ou adresse courriel) et ajoute le contact.
4. Un Toast message lui confirme l'ajout.

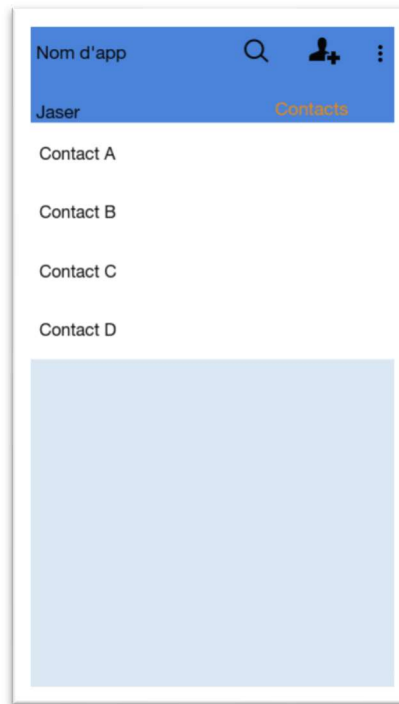


5. Les contacts seront listés

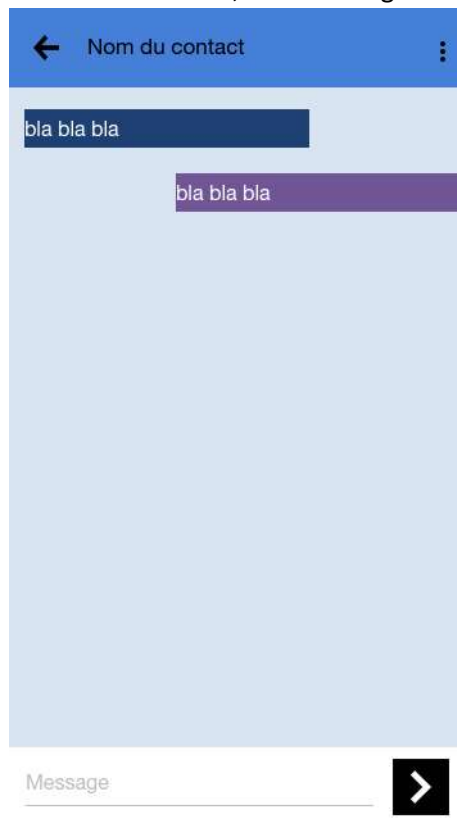


Conversation

1. L'utilisateur clique sur l'onglet contact
2. L'utilisateur clique sur un contact spécifique



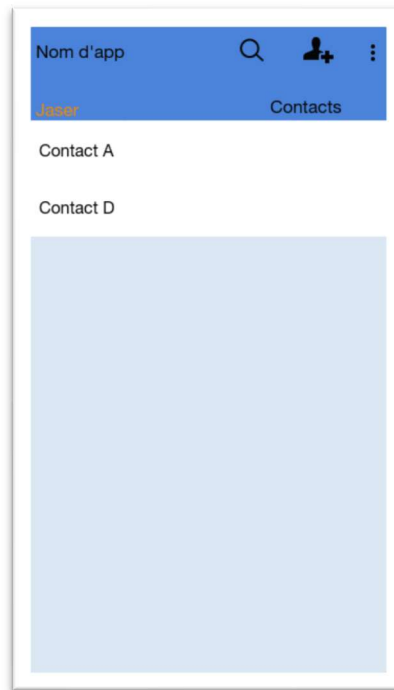
3. De que l'utilisateur a été cliqué sur le contact désiré, il est redirigé vers l'activité de conversation



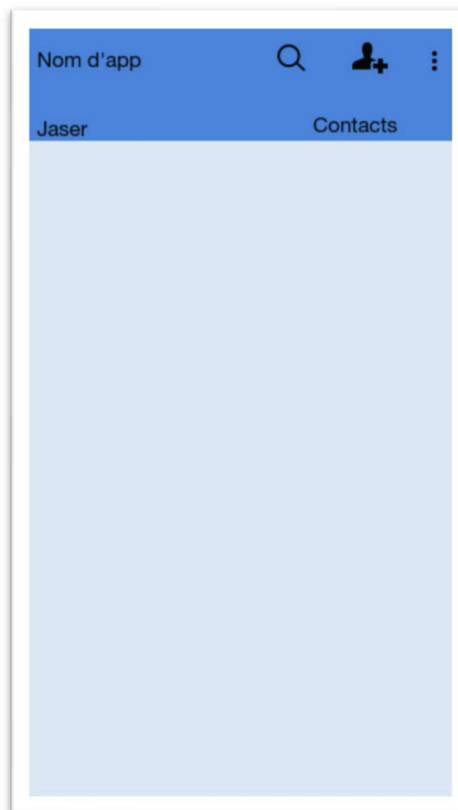
4. Les utilisateurs peuvent échanger des messages

Liste de Conversation

1. De que l'utilisateur est connecté à l'application, la page d'accueil sera la liste des conversations avec les contacts qu'il a contactée avant.



2. S'il n'y a pas de conversations, la page sera « en blanche »



3. En cliquant sur le bouton de retour sur le téléphone, ou sur la flèche à côté du nom de l'utilisateur qui a une conversation (fenêtre de conversation), l'utilisateur retournera à la liste des conversations.