Rapport de Projet : TakeTalk

Amar Hazem
Karlo Glavas
Hasnae ElGaad
Arezki Smail
Théodore Corbeau

Sous la direction de Monsieur Laurent Poligny

Université d'Evry Val d'Essonne

Sommaire

Introduction

Pour notre M2, trois notes nous seront attribuées. Une note théorique, une note de mémoire et une note de pratique. Cette dernière sera obtenue au travers de trois projets dont celui qui est traité dans ce rapport.

Il consiste à créer une application web en temps réel qui aura pour but de gérer le temps de parole des participants d'une réunion. Cela prendra la forme d'un forum privé ou chaque personne pourra se mettre en file d'attente pour parler du sujet qu'elle veut aborder.

Ce projet nous a été proposé par monsieur Poligny.

I. Analyse du sujet

1) Le problème

Après avoir fait un constat, monsieur Poligny s'est vu faire face à un besoin : Celui de gérer le temps de parole de ses réunions. Lors de réunions, les idées doivent être énoncées clairement, rapidement et de manière compréhensible pour que leur déroulement soit efficace.

Deux cas non désirables se posent alors :

-Une personne prenant la parole prend trop de temps pour s'exprimer, ce qui donne moins de temps aux autres participants pour parler et qui rends la réunion moins dynamique.

-Une personne ayant une idée pertinente ne trouve pas l'occasion de l'exprimer.

TakeTalk sera un système qui nous servira à remédier à ce problème de la manière qui va suivre.

2) Objectifs

Notre application va devoir fournir une interface unique accessible par tous les participants d'une réunion, sur laquelle ils peuvent interagir en temps réel. Chacun d'eux pourra se mettre en file d'attente pour pouvoir parler de son sujet. Ils pourront aussi sortir de la file s'ils le désirent.

Un de ces participants se démarquera des autres en tant qu'animateur et aura droit à des possibilités supplémentaires. Parmi elles, il pourra choisir d'arrêter ou de lancer un timer, ainsi que de passer à la personne suivante.

Un système d'invitation par mail sera mis en place pour cet utilisateur uniquement.

Pour que la solution soit vraiment efficace, chaque participant d'une réunion doit être capable de l'utiliser sur leur propre appareil, quel qu'il soit (smartphone, tablette, PC).

3) Contraintes

Pour mener à bien notre projet, certaines contraintes d'organisation seront mises en place.

Premièrement, nous utiliserons une méthode agile, pour encadrer notre travail. Celle que nous avons choisie est Scrum, que nous avons simplifiée pour s'accorder à la taille modeste de notre projet. Le déroulement de notre projet sera organisé en sprints d'une semaine qui dureront plus longtemps si la fonctionnalité désirée n'est pas atteinte.

Deuxièmement, GitHub sera aussi utilisé pour avoir un dépôt partagé par tous les membres de l'équipe et pour nous fournir une traçabilité au niveau de l'avancement du projet et des futures modifications.

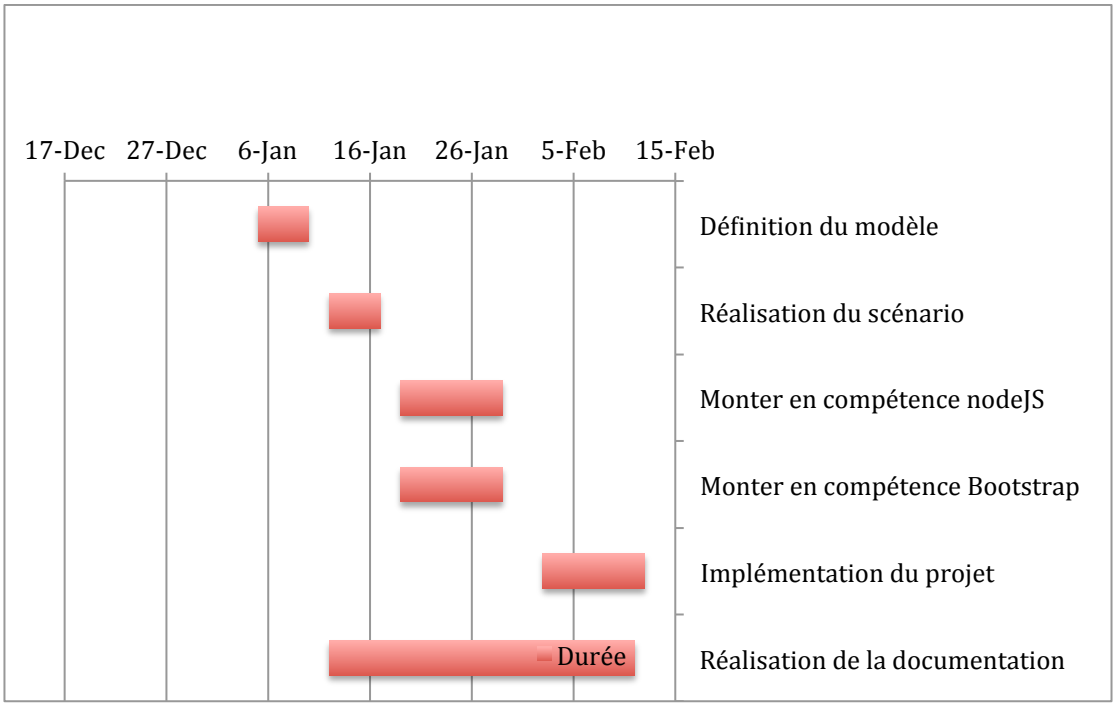
4) Organisation

Pour avoir une répartition des tâches efficace, les rôles ont été répartis ainsi :

- Un concepteur : C'est lui qui a désigné les IHM ainsi que les scénarios et la structure de la base de donnée.
- Trois développeurs : Un développeur principal s'occupe de tout le code et deux autres sont en soutien pour les parties difficiles à débuguer et les parties non critiques. Lorsqu'un problème ne semble pas se résoudre, deux personnes se mettent ensemble pour avoir deux points de vue différents et accélérer la résolution.
- Une rédactrice : Elle se concentre sur la partie rédaction de rapport, présentation, tutoriel.

Malgré cette répartition des rôles nous nous sommes réservé le droit de participer à des tâches qui ne nous étaient pas assignées ou bien de demander de l'aide aux autres personnes du groupe.

Un planning de projet a été réalisé pour optimiser le temps de travail. Il n'a pas été entièrement tenu mais il a été pratique pour forcer l'avancement :



II. Conception

1) Choix des technologies

A. Bootstrap

Il était très important que l'application soit utilisable sur n'importe quel appareil. Mais coder le HTML/CSS en prenant compte tous les types d'appareil ainsi que tous les navigateurs nous aurait demandé bien plus de temps que nous en disposions. L'utilisation d'un framework CSS était nécessaire et nous nous sommes arrêté sur Bootstrap. Bien que Foundation offre les mêmes services, Bootstrap semble plus fourni en documentation et plus soutenu par une communauté qui est susceptible de nous aider dans notre démarche.

B. NodeJS

Pour le choix du langage, la contrainte du temps réel était notre moyen de décision. Plusieurs solutions telles qu'ASP.net se présentaient mais NodeJS a retenu notre attention. Ce n'est pas à proprement parlé un langage mais une plateforme usant de javascript du coté serveur. Ce choix est motivé par la rapidité présentée par nodeJS, le fait qu'il soit non-bloquant et sa prédisposition à l'implémentation d'applications temps réel.

C. Meteor

NodeJS est très rapide mais a l'inconvénient d'être très bas niveau en contre-parti. Il y a beaucoup de configuration à effectuer et trop d'éléments à prendre en compte. Nous avons donc opté pour l'un des très nombreux framework javascript : Meteor. Ce choix est motivé par le fait qu'il permet de coder des applications single-page en temps réel de manière très simple. De plus nous avons découvert des services impressionnants au fur et à mesure du développement. Il nous a permis entre autre de déployer le site en ligne à travers une simple ligne de commande.

D. Anciens choix

Au départ, l'usage d'un framework lourd nodeJs ne nous semblait pas nécessaire. Le premier choix s'était donc porté sur expressjs. C'est un framework très léger qui s'apprend très vite mais dont on atteint vite les limites. De plus socket.io est assez difficile à appréhender pour faire du temps réel.

2) Scénarios

Voici les scénarios de cas d'utilisation
Ouverture d'une réunion :
- Cliquer sur le bouton « Open a meeting »
- Remplir tous les champs correctement
- Cliquer sur « Create »

Rejoindre une session :
- Cliquer sur le lien dans le mail reçu
- Si vous rejoignez cette réunion pour la première fois
  - Entrer votre nom
  - Cliquer sur « Join »

Parler :
- Si vous n'êtes pas en attente
  - Cliquer sur « Talk »
  - Remplir tous les champs correctement
  - Cliquer sur « Lineup »

Annuler speech :
- Si vous êtes en attente
  - Cliquer sur « Cancel talk »

Démarrer le timer :
- S'il y a au moins un speech en attente
  - Cliquer sur « Proceed »

Arrêter le timer :
- S'il y a un speech en cours
  - Cliquer sur « Wait »

Passer au speech suivant :
- S'il y a au moins un speech en attente ou un speech en cour
  - Cliquer sur next

Inviter des participants
- Cliquer sur « Invite participants »
  - Cliquer sur « Invite »

Note : Vous trouverez les IHM illustrant ces cas d'utilisation en annexe de ce rapport.

4) Objets définis

Notre application contient une base de données NoSql MiniMongoDB qui est bien pratique pour stocker les « objets » suivants dans des collections :

Meeting	User	Speech
• _id : int • name : string • status : string	• name : string • email : string • type : string • status : string • meeting : int	• _id : id • subjet : string • timeLeft : int • time : int • status : string • user : int • meeting : int

III. Implémentation

1) Installer TakeTalk pour développer

Il faut au préalable avoir installé

- NodeJs (A télécharger ici : Nodejs.org/download)
- Meteor (Lancer la commande : curl https://install.meteor.com/ | sh)

Il faut ensuite

- Forker et cloner le projet (github.com/wuanshu/taketalk)
- Se diriger dans le répertoire du projet
- Lancer la commande : meteor, pour déployer le serveur local
- Aller à l'adresse localhost:3000 pour voir le résultat

Pour tester l'application, il suffit d'aller à l'adresse suivante :
Taketalk.meteor.com

2) Structure du projet et de Meteor

Les dossiers de TakeTalk sont structurés ainsi :

Client
    Javascripts
    Stylesheets
    Views
        Application
Collections
Lib
Public
Server

Tout ce qui se trouve dans Client est exécuté par le client et tout ce qu'il a dans serveur est exécuté par le serveur. Tout le reste peut être exécuté par les deux.

Dans Lib, on trouve le document Router.js qui permet de diriger les adresses entrée vers la page HTML qui correspond et qui traite les données des collections pour les envoyer au moteur de template qui pourra générer les pages.

Dans javascripts, on trouve events.js qui traite tous les évènements triés par page. A chaque évènement correspond une action sur les collections. Cela va les modifier et Meteor recrée les pages en temps réel.

3) Fonctionnalités supplémentaires de Meteor

En plus de gérer automatiquement le temps réel sous forme de single-page app, Meteor nous a offert plusieurs services qui ont été appréciables :

- L'hébergement sur leurs serveurs est gratuit et ne demande qu'une inscription. De plus elle se fait de la manière la plus simple qui soit : Commande : meteor deploy taketalk.meteor.com

- Lorsque les envois de mails ne sont pas configurés, Meteor prévoie par défaut un couplage avec le service MailGun lorsque le site est déployé.
- Meteor nous a permis de créer des applications iOS et Android simplement en lançant une commande :
  Meteor run ios|android

Tout cela nous donne un environnement de test hyper réactif et en condition d'utilisation normal.

4) Difficultés rencontrées

Les difficultés rencontrées lors de ce projet sont principalement de l'ordre de l'implémentation. Le concept de l'application est très compréhensible ce qui nous a grandement facilité la tache pour la phase de conception.

Comme présenté au début du rapport, NodeJs est très bas niveau et demande de configurer plusieurs paramètres. Le choix d'expressjs couplé à socket.io nous a fait perdre du temps. Meteor a permis de gommer beaucoup de détails en nous faisant nous concentrer uniquement sur les évènements et le routing. En effet, Meteor re-génére les interfaces de tous les clients à chaque fois qu'une collection a été modifiée. De plus les objets du DOM sont facilement récupérables via les évènements.

5) Améliorations possibles

TakeTalk est actuellement dans une version utilisable mais peut bien sûr subir des améliorations pour rendre l'expérience bien plus pratique. En voici quelques-unes :

- Un système de drag and drop pour que l'animateur puisse changer l'ordre des participants dans la file d'attente.
- Une croix sur chaque speech pour que l'animateur puisse les supprimer.
- Un menu de configuration pour que l'animateur puisse gérer les fourchettes de temps admises.
- Un bouton pour modifier le nom du meeting.
- Une barre de progression pour suivre le chargement du site.
- Afficher la liste des speechs passés
- Ajouter un système de tags sur les speechs. Par exemple « important », « facultatif »…
- Traduire l'application et afficher la langue selon la localisation tout en donnant la possibilité de changer.
- Donner la possibilité de créer un compte pour voir son historique.

Conclusion

Après ces deux mois de conception et de programmation, nous estimons avoir atteint nos objectifs exceptée la possibilité d'ordonner les passages. Nous avons un système très évolutif avec un potentiel suffisant pour devenir un outil populaire.

Au niveau technologique, l'apprentissage de NodeJS a été très bénéfique surtout avec l'usage de meteor.

Au niveau organisationnel, bien que notre formation avait des failles, elle aura eu le mérite de nous avoir fait essayer une nouvelle configuration (répartition des tâches par domaine).

Bibliographie

Sites des outils :

- Nodejs.org
- www.meteor.com
- getbootstrap.com
- www.mongodb.org

Sites de tutoriels

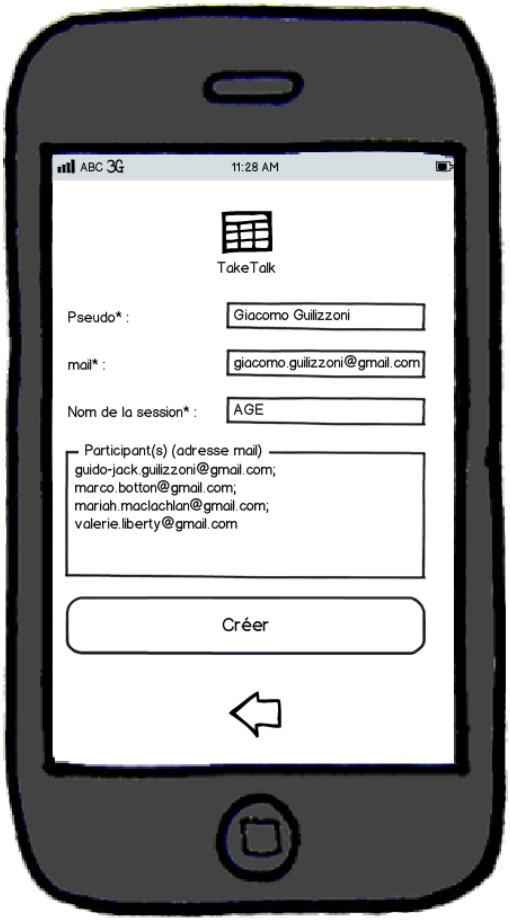
- openclassroom.com
- stackoverflow.com

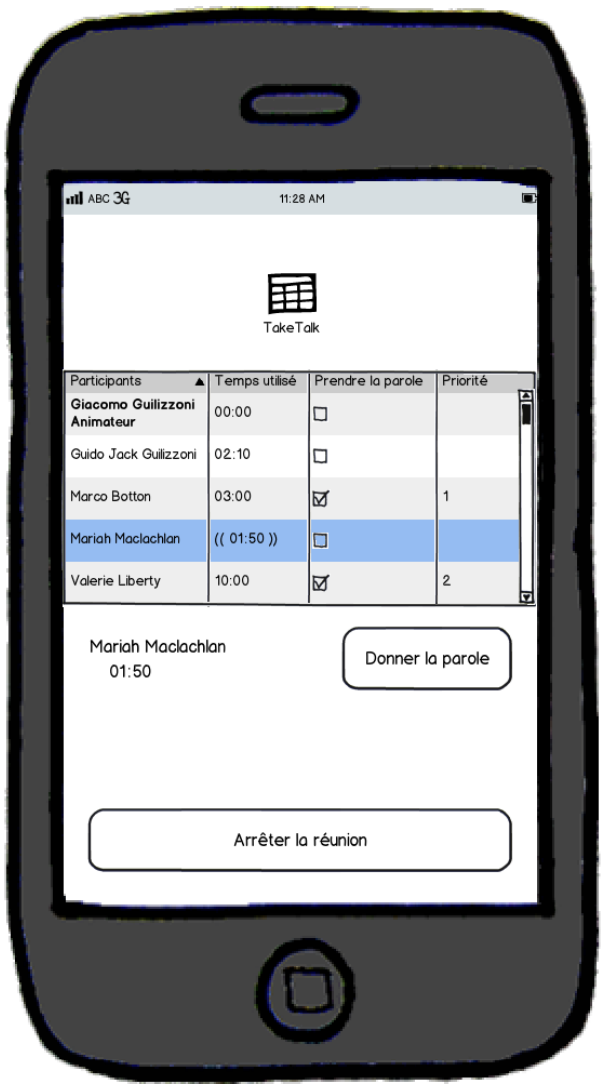
Sites des outils

- github.com
- www.scrum.org

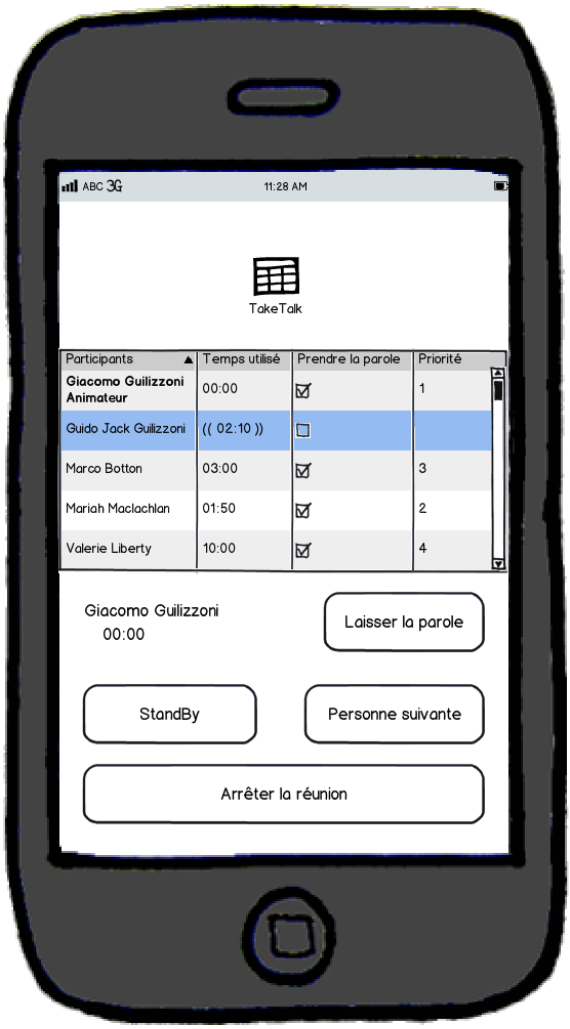
Annexes

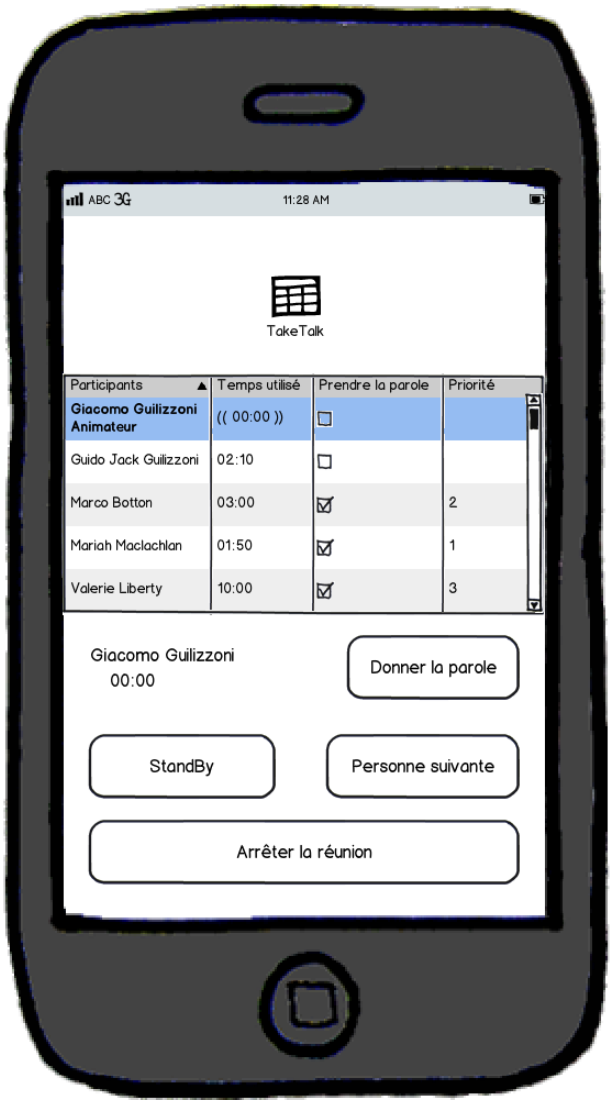
- Conception de l'IHM de TakeTalk

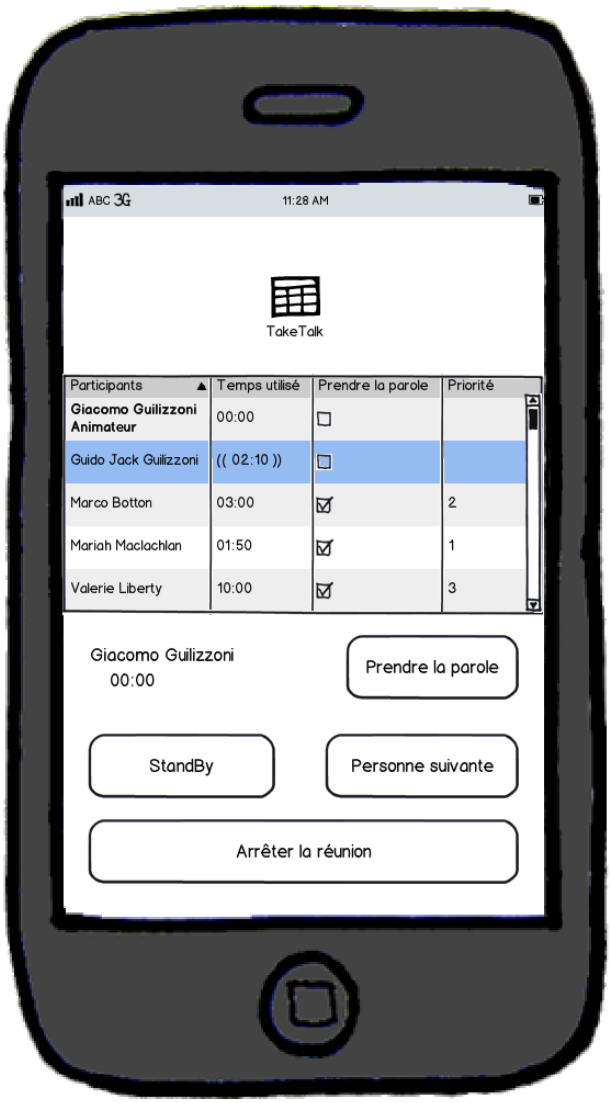



ABC 3G
11:28 AM
TakeTalk
Participants
Temps utilisé
Prendre la parole
Priorité
Giacomo Guilizzoni Animateur
00:00
Guido Jack Guilizzoni
02:10
Marco Botton
03:00
1
Mariah Maclachlan
((01:50))
Valerie Liberty
10:00
2
Mariah Maclachlan
01:50
Donner la parole
Arrêter la réunion


ABC 3G
11:28 AM
TakeTalk
Participants
Temps utilisé
Prendre la parole
Priorité
Giacomo Guilizzoni Animateur
00:00
Guido Jack Guilizzoni
01:50
Marco Botton
((03:00))
Mariah Maclachlan
02:10
1
Valerie Liberty
10:00
2
Marco Botton
03:00
Donner la parole
Quitter réunion


ABC 3G
11:28 AM
TakeTalk
Participants
Temps utilisé
Prendre la parole
Priorité
Giacomo Guilizzoni Animateur
00:00
1
Guido Jack Guilizzoni
((02:10))
Marco Botton
03:00
3
Mariah Maclachlan
01:50
2
Valerie Liberty
10:00
4
Giacomo Guilizzoni
00:00
Laisser la parole
StandBy
Personne suivante
Arrêter la réunion


ABC 3G
11:28 AM
TakeTalk
Participants
Temps utilisé
Prendre la parole
Priorité
Giacomo Guilizzoni Animateur
((00:00))
Guido Jack Guilizzoni
02:10
Marco Botton
03:00
2
Mariah Maclachlan
01:50
1
Valerie Liberty
10:00
3
Giacomo Guilizzoni
00:00
Donner la parole
StandBy
Personne suivante
Arrêter la réunion


ABC 3G
11:28 AM
TakeTalk
Participants
Temps utilisé
Prendre la parole
Priorité
Giacomo Guilizzoni
Animateur
00:00
Guido Jack Guilizzoni
((02:10))
Marco Botton
03:00
2
Mariah Maclachlan
01:50
1
Valerie Liberty
10:00
3
Giacomo Guilizzoni
00:00
Prendre la parole
StandBy
Personne suivante
Arrêter la réunion

1 Rapport de Projet : TakeTalk