1) Présentation du projet :

Le but de ce projet est de créer une application Android de gestionnaire de rendez-vous.

Un rendez-vous doit être caractérisé par un titre, une date, un horaire, une adresse, un contact ainsi que son numéro de téléphone.

L'application doit faire apparaître tous les rendez-vous créés dans une liste où apparaissent les différentes informations citées ci-dessus.

Il doit donc être possible d'ajouter des rendez-vous, mais aussi de les modifier ou de les supprimer.

Dans l'éditeur de rendez-vous, il doit être possible de changer la date et l'heure en utilisant des pickers, d'aller chercher l'adresse sur google maps, d'appeler le numéro de téléphone renseigné, et de choisir un contact parmi la liste de contacts du téléphone (optionnel).
Il doit être également possible de partager les informations d'un rendez-vous.

Un système de notifications doit également prévenir l'utilisateur que son rendez-vous approche, selon ses préférences.
Il doit être également capable de modifier ses préférences, pour changer par exemple les couleurs.

Enfin, l'application doit pouvoir être affichée en portrait ou en paysage, et dans le cas du paysage, afficher la liste et l'éditeur en même temps.

2) Fonctionnalités :

- Connection Database / Liste items : Réussie
- Ajout de RDV : Réussie
- Edition de RDV : Réussie
- Suppression de RDV : Réussie
- Portrait/Landscape : Réussie, mais sans Fragments.
- Langages Français/Anglais : Réussie
- Gestion des préférences : Non implémenté.
- Alarmes / Notifications : Non implémenté par manque de temps / Incompréhension de l'implémentation dans le projet.
- 2 Thèmes : Non implémenté.

- Possibilité d'utiliser Google Maps depuis l'application : Réussie
- Possibilité de partager les données de l'application : Non implémenté
- Possibilité d'appeler un numéro de téléphone renseigné : Réussie, mais avec un léger défaut (voir passage concerné).

3) Contenu :

A) Classe RDV :

La première chose nécessaire était de créer une classe d'objets rendez-vous (nommés RDV pour la suite).

Elle se compose des attributs suivants :

```
public class RDV implements Parcelable {
    private long id;
    private String title;
    private String date;
    private String time;
    private String address;
    private String contact;
    private int phoneNum;
    private boolean state;
```

L'id étant géré par la base de données, nous en parlerons le moment venu.
Le booléen state quant à lui définit si un rendez-vous a été effectué ou non.

La classe contient trois constructeurs :

- Un renseignant en paramètres tous les attributs sauf id, qui sert à la création d'un RDV (pour rappel, id est généré par la base de données).
- Un autre renseignant tous les attributs dont id, qui sert à la modification de RDV
- Un autre renseignant un Parcel, qui permet de créer un objet RDV via lecture de ce Parcel, qui permet donc de passer des RDV d'une activité à l'autre.

Il existe donc au sein de la classe RDV les méthodes permettant d'écrire de tels Parcels via les objets RDV.

Enfin, chaque attribut possède un getter et un setter pour pouvoir être récupéré/modifié par les méthodes des autres classes.

B) Classe DataBase :

_____Cette classe hérite de la classe SQLiteOpenHelper et nous sert à créer une base de données pour l'application qui va stocker tous les rendez-vous que l'on va créer.

Comme elle fonctionne via des requêtes SQLite, nous utilisons des paramètres sous forme de static String dans cette classe.

C'est cette classe notamment qui gère les id propres à chaque rendez-vous : ceux-ci s'obtiennent de façon automatique à la création d'un RDV dans la dataBase, en récupérant la valeur du dernier rendez-vous renseigné plus un.
Ce point nous a posé problème lors de l'affichage dans les xml, car lors de la création d'un RDV, ce dernier ne possède pas encore d'id, ce qui amène à des NullPointerException à l'exécution. Ceci a néanmoins été corrigé.

La classe contient des méthodes et des procédures pour ajouter, modifier et supprimer des RDV dans la database, en prenant un objet RDV en paramètre.

Elle possède également une méthode retournant un Cursor contenant la totalité des RDVs présents dans la dataBase, ce qui permet de les afficher.

Cette dernière méthode nous a pendant un temps posé difficulté pour justement lier ce Cursor à une list view qui permettait de l'afficher, mais cela venait plus d'une incompréhension de son fonctionnement qu'autre chose.

C) Classe MainActivity :

C'est la classe principale instanciée au lancement de l'application.
Elle comporte donc un attribut DataBase qui va contenir tous nos RDVs, ainsi que la ListView permettant d'afficher son contenu.

A la création de l'activité, on ouvre cette DataBase et on charge ses données pour les injecter dans la ListView.

Si un Item de cette liste est cliqué, on ouvre l'éditeur de RDV. Ceci a posé problème car il fallait faire une distinction à l'ouverture de l'éditeur de RDV entre l'ajout et la modification de RDVs, afin de ne pas ajouter un RDV dans la DataBase en cas de modification. Cela nous a amené à créer un Booléen dans la classe de l'éditeur, dont nous parlerons plus tard, et ce booléen est alors passé à false via un intent lorsque l'on clique sur un Item de la liste, pour bien spécifier qu'on modifie un RDV.

Il est également possible de maintenir le clic sur un Item pour afficher un menu déroulant dont la seule option est de supprimer le RDV en question.

Enfin, sur le menu supérieur, un bouton “+” permet d’ajouter un RDV (et via un Intent, lance l’activité de l’éditeur de RDV, en spécifiant cette fois le booléen à true).

D) Classe RDVEditor :

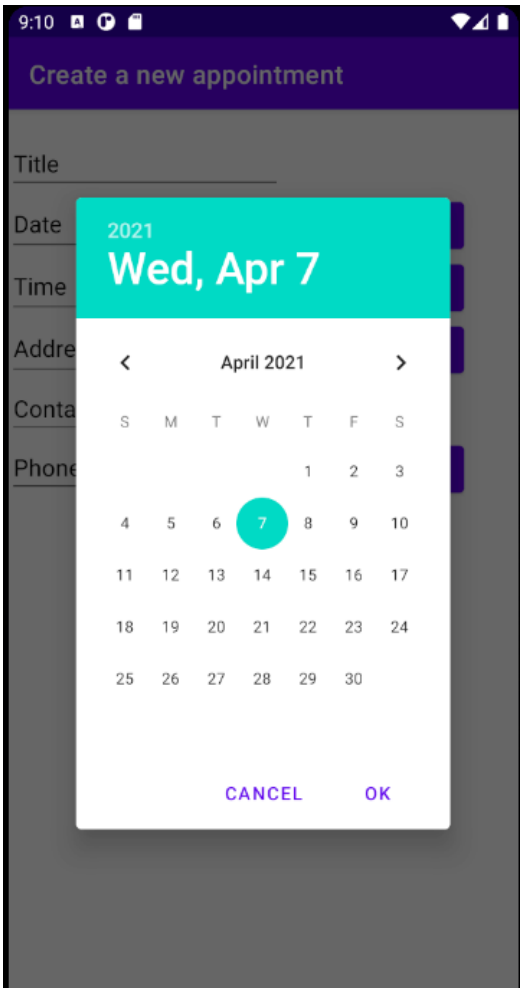
Il s’agit de la classe permettant d’éditer un nouveau RDV, ou d’en modifier un.

Elle contient un grand nombre d’attributs, puisque c’est là qu’on renseigne les attributs de chaque RDV.

Entre autres, comme nous utilisons des boutons pour remplir certains attributs, cela complexifie un peu plus cette classe.

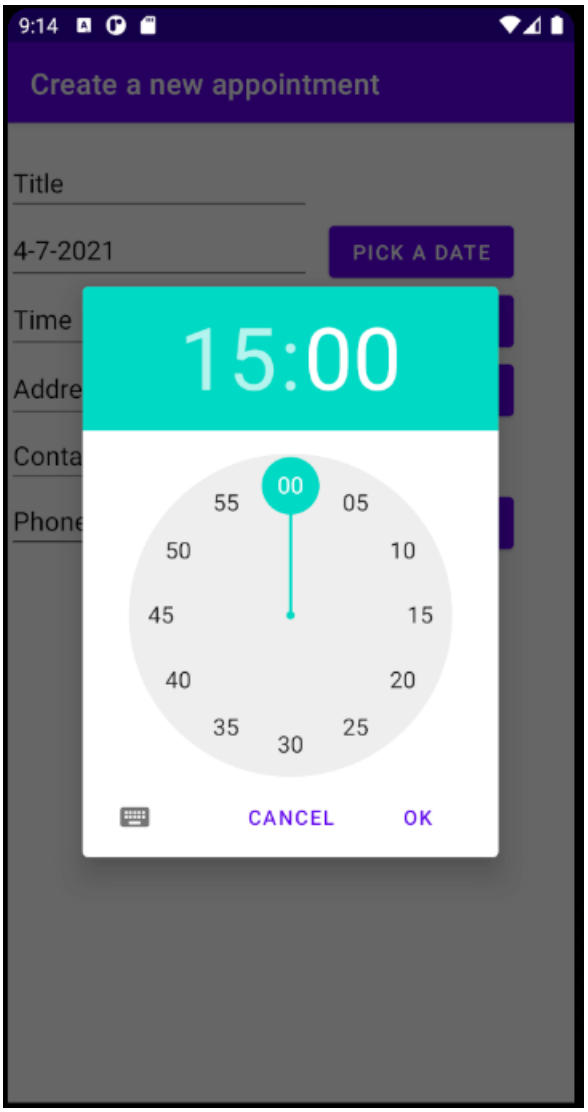
Cette activité contient un EditText pour chaque attribut modifiable d’un RDV (c’est-à-dire tous sauf l’id).

Pour la date et l’heure, on utilise des boutons permettant de les sélectionner depuis des date et time Pickers.



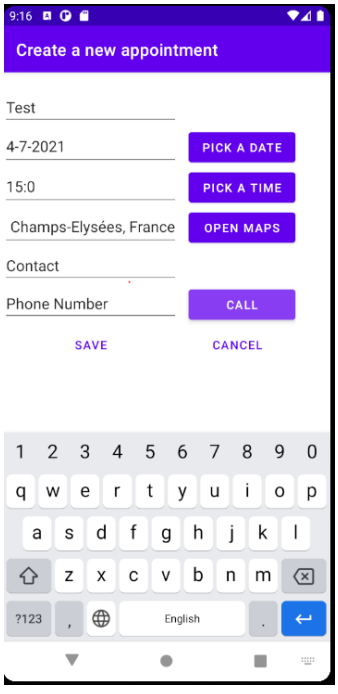
Ainsi, en cliquant sur le bouton “Pick a Date”, on ouvre un Date Picker (instancié par la classe DatePickerFragment, qui permet de récupérer le jour, le mois et l’année sélectionnés et de le passer à l’EditText correspondant).

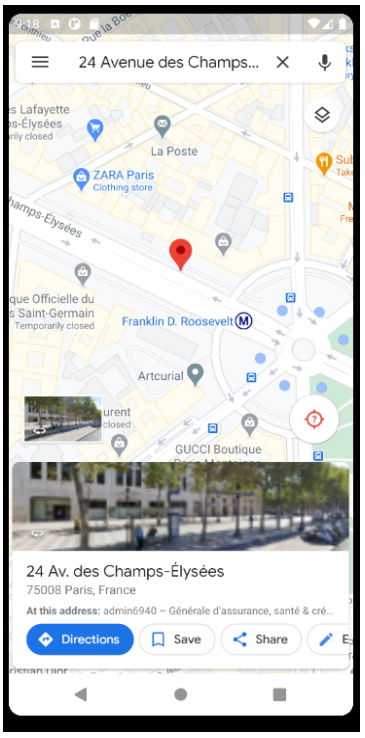
Cliquer sur OK permet de définir le jour sélectionné comme attribut du RDV (et il apparaît dans le Edit Text correspondant).



De même avec l'heure, il est possible de la sélectionner à l'aide d'un TimePickerFragment, qui fonctionne sur le même principe.

En renseignant une Adresse dans le champ correspondant, il est possible de cliquer sur le bouton "Open Maps" pour visualiser sa localisation dans Google Maps :



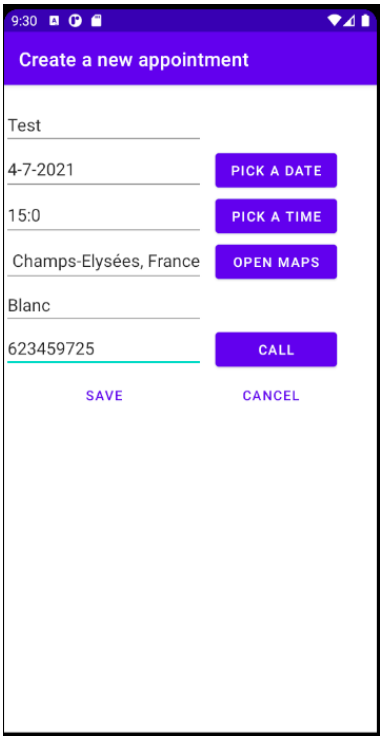


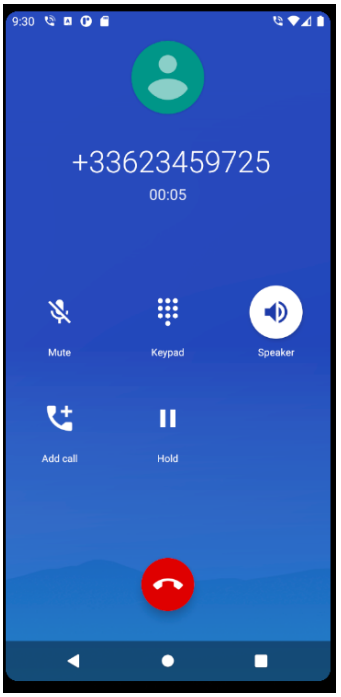
Les champs Titre et Contact doivent être rentrés “à la main”.

Nous avions tenté de faire un Picker de Contact, mais sans succès, ayant rencontré des crashs de l’application sans en comprendre la raison, car nous avions pourtant donné les permissions à l’accès aux contacts du téléphone au préalable. Nous avons soupçonné que notre Picker ne donnait pas le format adéquat pour l’EditText, et que c’était une raison possible du crash de l’application.
Par conséquent, nous avons retiré toute trace de ce bouton.

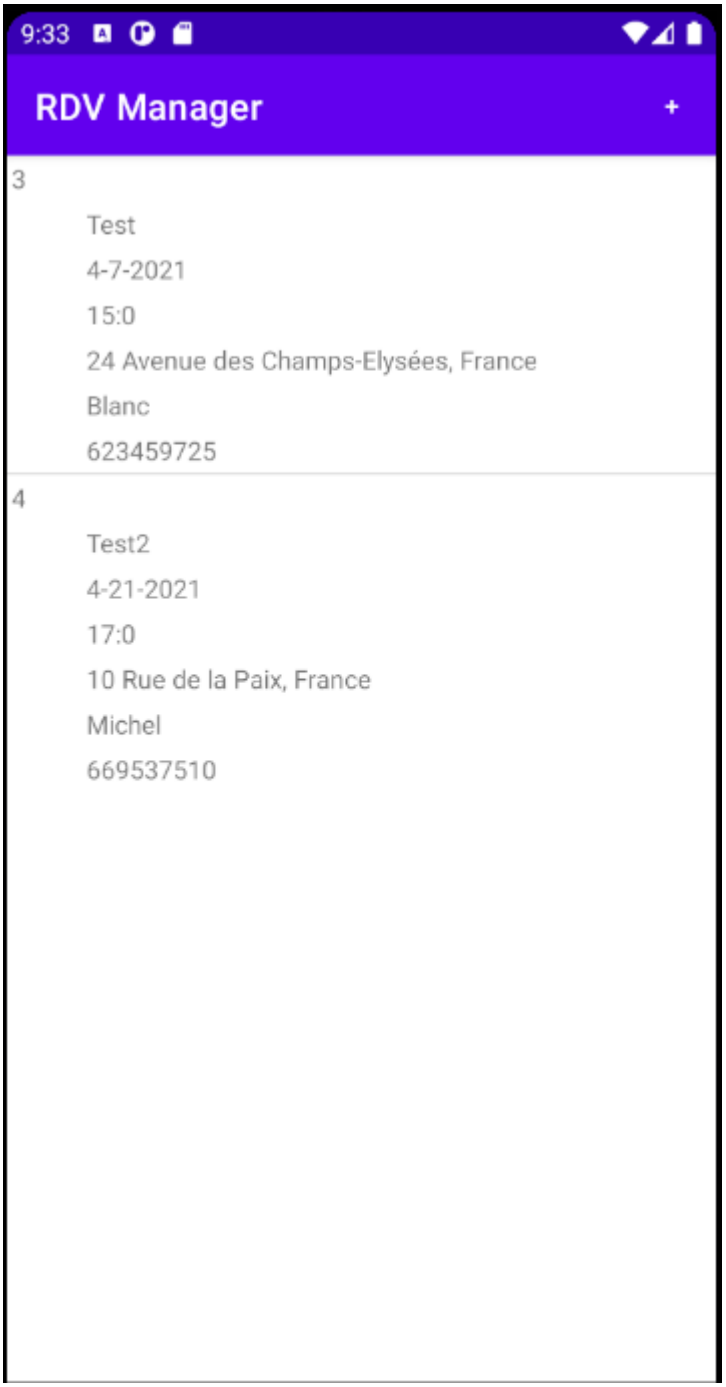
Il faut aussi renseigner l’EditText Phone Number avec un numéro de téléphone. Le bouton associé permet d’appeler le numéro en question.
Nous l’avons paramétré pour qu’il puisse appeler un numéro français (avec l’indicateur téléphonique +33). L’appui sur le bouton entraîne une demande de permission pour avoir accès à la fonction “téléphone” de l’appareil.
Cependant, cela entraîne un crash de l’appli avant que l’utilisateur n’ait eu le temps de donner cette permission.
Mais une fois qu’il l’a donnée, il est possible d’appeler.

Le numéro doit donc être renseigné comme ceci :



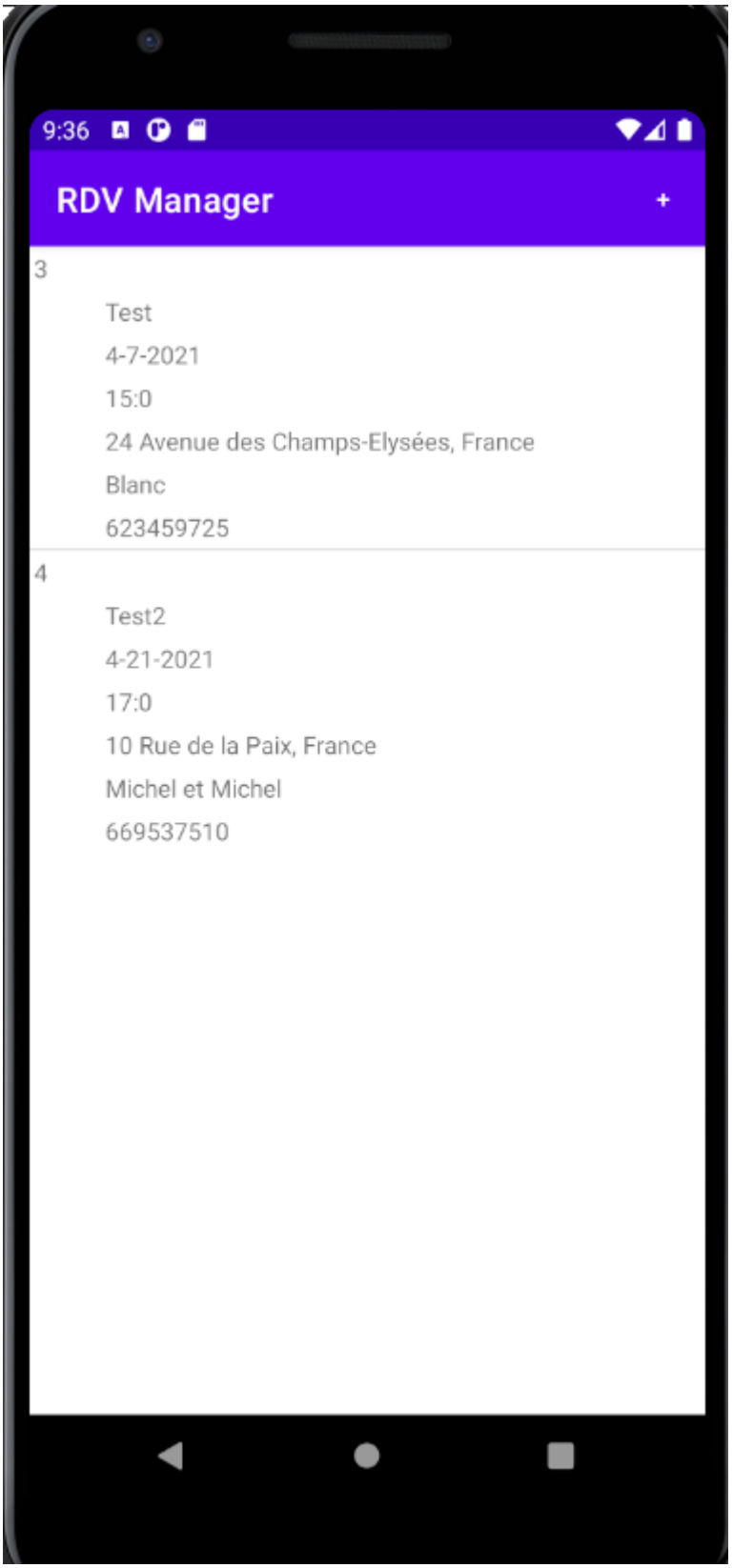


Enfin, le bouton Save permet de sauvegarder le RDV dans la DataBase, et il sera donc affiché dans la liste du main activity :



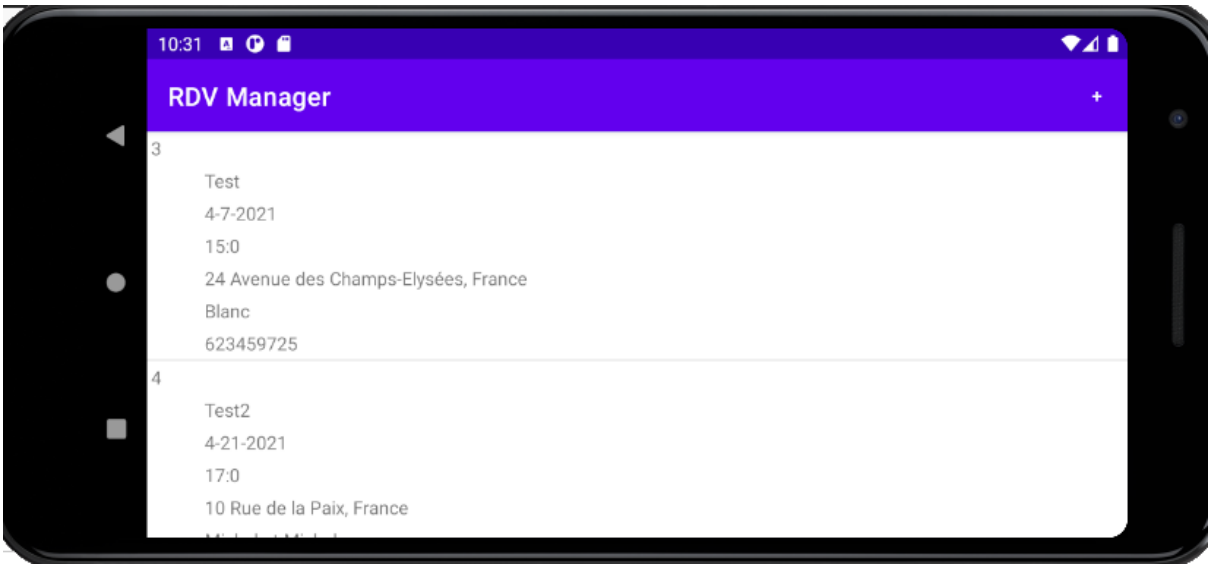
A l'inverse, appuyer sur le bouton Cancel permet de revenir en arrière sans créer de RDV.

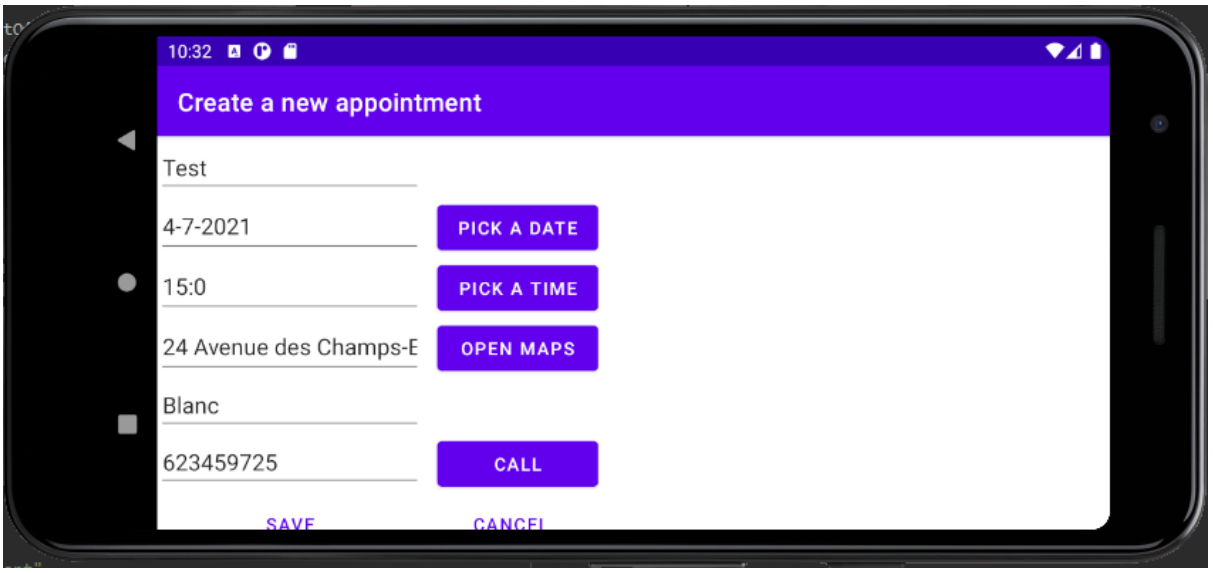
Cliquer sur un RDV ouvre le même menu que précédemment, mais la sauvegarde ne crée pas un nouveau RDV, elle ne fait que le modifier :



On constate ici que le nom du contact a changé, mais que l'id est resté identique, prouvant bien qu'il s'agit d'un RDV modifié et non d'un nouveau.

Enfin, l'application fonctionne en mode paysage, mais n'utilise pas de fragments :





Il est possible de scroll l'application.

Le langage peut également être changé dans les paramètres du téléphone, les deux seuls langages disponibles étant le français et l'anglais :