# TASK PLANNER

Vlad Ionescu, 342 C3
Andrei Ciocan, 342 C3

**Data:** 18.03.2015
**Versiune document:** 0.0.0.0.1

1. **Descriere proiect**

- **Rezumat**

**Task Planner** este un proiect destinat utilizatorilor de smartphone-uri Android, cu scopul de a usura implinirea activitatilor zilnice prin ordonarea lor optima, in functie de diferite constrangeri: orar, meteo, trafic. Este demonstrat faptul ca o persoana iroseste foarte mult timp dintr-o zi intre activitatile pe care trebuie sa le indeplineasca: in trafic, in ferestrele de timp goale, facand un drum de mai multe ori etc. **Task Planner** este o aplicatie care va crea un calendar optim pentru o persoana, minimizand timpii morti, dar in acelasi timp pastrand un echilibru intre munca si pauza.

- **Detaliat**

Dandu-se o lista de activitati pe care o persoana trebuie sa le indeplineasca pe parcursul unei zile, alaturi de o serie de constrangeri, **Task Planner** va crea un program optim (care va minimiza timpul total de indeplinire a obiectivelor + timpul petrecut intre locatii), satisfacand toate constrangerile si luand in considerare si alti factori externi, independenti de utilizator, cum ar fi: conditiile meteo, conditiile de trafic etc.

Aplicatia va fi disponibila in doua versiuni: **gratis** si **platit**. Versiunea platita va prezenta cateva imbunatatiri fata de versiunea gratis, adresandu-se in special companiilor si angajatilor. In continuare se afla o lista cu feature-uri (specificandu-se la sfarsitul fiecareia pentru care versiune este disponibila) pe care aplicatia **Task Planner** le va implementa:

1. Utilizatorul va introduce activitatile pe care vrea sa le indeplineasca in acea zi. O astfel de activitate vine si cu o lista de constrangeri specifice:

a. de timp:

- activitatea trebuie indeplinita obligatoriu intre orele X si Y (exemplu: curs la facultate intre orele 14:00 si 17:00) [free].
- activitatea dureaza T ore si poate fi indeplinita oricand in intervalul orar X si Y (exemplu: o vizita de 1 ora dimineata, intre orele 10:00 si 14:00) [free].
- nicio restrictie sau detalii necunoscute inca (activitatea nu se stie cat va dura) - in acest caz, se va incerca plasarea ei in ferestre de timp libere mai largi [free].

b. de locatie:
- activitatea X trebuie sa fie indeplinita in locatia Y [free].
- activitatea X trebuie indeplinita, indiferent de locatie (exemplu: userul trebuie neaparat sa ajunga la o sucursala a Bancii Transilvania, indiferent unde). In acest caz, in functie de ordonarea activitatilor si locatia precedenta, va fi aleasa o locatie optima a sucursalei in care utilizatorul va fi trimis [free].

c. in functie de constrangerile altor utilizatori:
- calendarele a doi utilizatori **Task Planner** pot fi sincronizate (exemplu: utilizatorul X si Y trebuie sa participe la o sedinta impreuna). Astfel, celelalte activitati vor fi setate in asa fel incat activitatea comuna sa fie planificata in acelasi timp [paid].

2. Activitatile vor fi planificate in functie de conditiile meteo din acea zi. Ele pot fi replanificate in timpul zilei (activitatile care au mai ramas), in functie de conditiile de trafic / mijloace de transport (pentru deplasarile care nu pot fi facute pe jos).

3. Poate fi setat ca inainte cu X minute de activitatea urmatoare sa se primeasca o notificare. Numarul X va fi determinat in functie de conditiile mediului, pentru a nu oferi o notificare fixa (exemplu: traficul este foarte aglomerat, notificarea trebuie primita mai din timp).

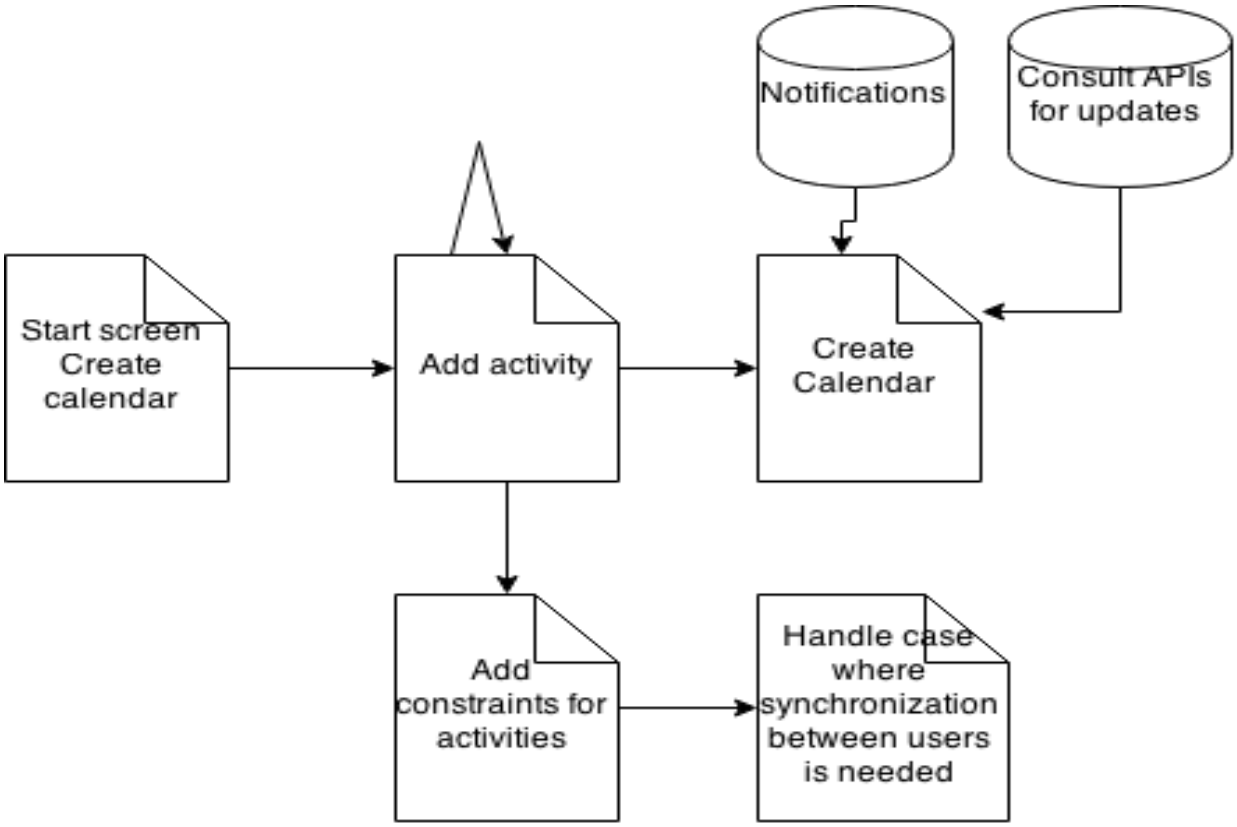
## 2. Tehnologii folosite

1. API-ul de weather, pentru a afla conditiile meteo: http://openweathermap.org/api.
2. API-ul pentru trafic, se va folosi Google APIs: https://developers.google.com/maps/documentation/javascript/trafficlayer
3. API-ul pentru a afla/cauta locatia unui obiectiv, se vor folosi tot Google APIs: https://developers.google.com/maps/
4. Android SDK
5. Baza de date MySQL stocata pe un server pentru a retine userii si interactiunile dintre calendarele lor
6. Algoritmi pe grafuri (exemplu: ciclu hamiltonian de cost minim)
7. Alte framework-uri decise pe parcurs, pentru a usura munca si a minimiza timpul alocat development-ului (exemplu: Hibernate pentru interactiunea cu BD etc.)

## 3. Workflow-ul aplicatiei

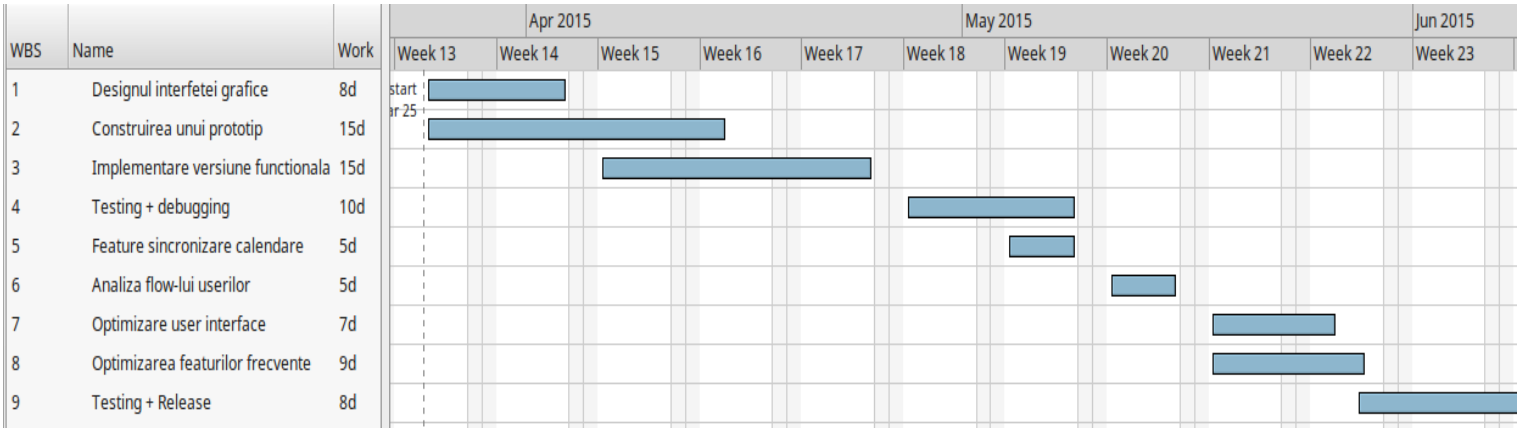
Aplicatia va trebui sa respecte urmatorul workflow, pentru a pastra coerenta design-ului si a oferi utilizatorilor un program placut vederii si cu care este usor de interactionat. La nivel de cod, aplicatia va fi impartita in diferite module (de decis) si se va implementa in Java.

Diagrama workflow-ului:




## 4. Sarcini de lucru

Echipa, formata din 2 membri, Andrei Ciocan si Vlad Ionescu, si-a desemnat (dupa un lung consiliu) urmatoarele responsabilitati si au fost impartite in timp in felul urmator, pe durata a 12 saptamani.



Astfel, proiectul este constituit din 9 etape, dupa cum urmeaza:

Etapa 1.

Implementarea si designul unei interfete grafice cat mai intuitive pentru utilizatori.

Timp de lucru alocat : 8 zile.

Etapa 2.

Construirea unui prototip in care vom integra functiile minimale necesare aplicatiei in interfata grafica. Acest pas este necesar pentru o mai buna evaluare a proiectului, si eventual pentru a regandi anumite aspecte ale interfetei grafice.

Timp de lucru alocat : 15 zile.

Etapa 3.

Implementarea unei versiuni functionale. Aceasta etapa presupune implementarea tuturor functiilor pentru varianta free a aplicatiei.

Timp de lucru alocat : 15 zile.

Etapa 4.

Testarea functionalitatii produsului.

Timp de lucru alocat : 10 zile.

Etapa 5.

Implementarea optiunii de sincronizare intre calendarele utilizatorului. Acest feature este integrat in varianta pro a aplicatiei.

Timp de lucru alocat : 5  zile.

Etapa 6.

Analiza flow-ului userilor pentru a colecta date despre aplicatie. Astfel, putem face o analiza pentru a adauga scurtaturi(shortcuturi) si a usura flow-ul utilizatorilor.

Timp de lucru alocat : 5  zile.

Etapa 7.

Optimizarea interfetei cu utilizatorul in conformitate cu analiza flow-lui. De exemplu, in urma analizei observam ca majoritatea userilor folosesc un feature, deci il putem porni by default in momentul cand aplicatia este rulata.

Timp de lucru alocat : 5 zile.

Etapa 8.

Incercarea optimizarii featurilor mai frecvent utilizate de catre useri. Aceasta etapa presupune o analiza adanca a featurilor si incercarea implementarii unor algoritmi mai eficienti in backend.

Timp de lucru alocat : 9 zile.

Etapa 9.

Ultima etapa o reprezinta construirea de teste aditionale, teste de stres, ce se intampla cand utilizatorul ramane fara semnal si nu poate primi update-uri de trafic etc.

Timp de lucru alocat : 8 zile.