# Document de analiză a cerințelor clientului

## Scopul aplicației:

*Identificați care este scopul principal al aplicației pe care o veți dezvolta. Pentru construirea scopului, veți începe de la cerințele de proiect din Curs I, dar îmbinat cu specificul aplicației pe care vă propuneți să îl faceți.*

*Scopul principal : Facilitieaza comunicarea userului cu un dispozitiv pentru ingrijirea si udarea plantelor si faciliteaza comunicarea cu alte dispozitive (ex: timer).*

## Obiectivele aplicației:

*Definirea unui număr mic de obiective care acoperă cele mai importante perspective ale aplicației. Puteți începe de la cum ar arăta un proiect minim viabil (eventual pornind de la cerințele din curs). Ce valoare aduce clientului? Ce particularități de dezvoltare trebuie avute în vedere? Ce metrici de evaluare sunt relevante?*

*În general scopul și obiectivele se construiesc împreună; scopul fiind o privire de ansamblu a proiectului (și deci a obiectivelor) iar obiectivele încearcă să clarifice care sunt livrabilele, care împreună definesc scopul.*

*Sa preia date despre vreme/caldura/umiditate utilizand un API, sa le prelucreze si sa transmita anumite informatii si recomandari user-ului .*

*Sa poata efectua automat anumite actiuni in functie de datele primite si in functie de anumite intervale de timp.*

*Sa poata comunica printr-un HTTP sau un MQTT API.*

## Grupul țintă

*Cui îi adresați această aplicație? Care este profilul utilizatorului a cărui nevoie căutați să o satisfaceți? În special trebuie să surprindeți felurile diferite în care utilizatori diferiți folosesc același feature (de exemplu, cum folosește un gamer numpad-ul, și cum îl folosește un contabil) Folosiți User Stories.*

*Aceasta aplicatie se poate adresa  specialistilor in domeniul cresterii plantelor,dar grupul-tinta este reprezentat de oamenii care nu au o legatura cu domeniul, dar vor sa invete sa aiba grija de plantele lor din gradina si nu dispun de timp pentru a supraveghea plantele.*

*User Stories:*

*Pentru un user  specialist:*

*Userul vrea sa poata seta intervalul de timp in care sa fie udate plantele*

*Userul vrea sa primeasca o alerta in cazul in care nu se poate efectua udarea plantelor din cauza lipsei de apa*

*Pentru un user care nu are prea multe cunostiinte despre plante:*

*Userul vrea ca momentul udarii plantelor sa fie ales in functie de vreme*

*Userul vrea ca planta sa aiba in permanenta o  anumita cantitate de lumina, iar in cazul in care nu o are ar trebui sa se porneasca automat o sursa de lumina artificiala*

*Userul vrea ca planta sa aiba o anumita temperatura in locul in care are planta, in cazul contrar ar trebui sa aiba un sistem automat de incalzire setat pe o anumita temperatura.*

## Colectarea cerințelor

*Care sunt cerințele pe care userii din story-urile de mai sus le-ar cere? Care sunt cerințele de sistem care apar ca o consecință a cerințelor userului (performanța în anumiți parametrii; utilizarea anumitor resurse externe; utilizarea unui mod specific de dezvoltare ș.a.m.d.). Această zonă este bine completată dacă are un număr cât mai mare de cerințe (relevante), chiar dacă sunt prea multe, sau depășesc un pic dimensiunea proiectului pe care îl avem în vedere. Aici trebuie să înțelegem tot ce* ***s-ar putea*** *face.*

*Cerinte functionale:*

1) *sa  ofere un program default pentru setarea parametrilor de udare si ingrijire al plantelor -> Procesarea datelor*
2) *sa ofere optiunea de a configura intervalul  de timp pentru udarea plantelor->* Cerinte independente de datele din API
3) *sa  ofere optiunea de a configura temperatura ->* Cerinte independente de datele din API
4) *sa  dea o alerta in caz ca nu se poate efectua udarea plantelor ( nu mai este suficienta apa)*

   *->* Cerinte independente de datele din API

5) *sa dea o alerta in cazul in care apare o defectiune tehnica (ex: se ard becurile) - >* Cerinte independente de datele din API
6) *sa faca recomandari userul-ui in functie de vreme -> Procesarea datelor*
7) *sa primeasca umiditatea solului si sa decida daca sa porneasca apa -> Procesarea datelor*
8) *setarea parametrilor sa poata fi facuta pe un interval cat mai larg ->* Cerinte independente de datele din API
9) *setarea parametrilor individuala pentru fiecare planta ->* Cerinte independente de datele din API
10) *alerta pentru conditii meteo nefavorabile pentru plante (grindina, canicula) ->* Cerinte independente de datele din API

*Cerinte nefunctionale:*

- *Aplicatia trebuie sa consume un API de vreme  -> Comunicare*
- *Sa poata comunica printr-un protocol HTTP sau un MQTT  cu un API. -> Comunicare*
- *Toate functionalitatile vor avea asociate cate un unit test. -> Testare*
- *Aplicatia trebuie sa aiba Integration tests -> Testare*

## Interpretarea și prioritizarea cerințelor

*Dintre cerințele de mai sus vom interpreta și prioritiza cerințele.*

*1. Label-uiți cerințele funcționale / non-funcționale. Cerințele funcționale sunt cele care îndeplinesc o nevoie care a reieșit dintr-un user story, și răspund la întrebarea ce trebuie aplicația să facă. Cerințele nonfuncționale sunt cele care descriu calitățile de sistem, și răspund întrebărilor de tipul cum trebuie să fie un anumit feature sau aplicația cu totul? Acest label-ing e suficient să îl faceți în documentul de analiză, nu e necesar să îl cărați pe mai departe.*

*2. Gruparea cerințelor*

*Identificați criterii de gruparea cerințelor care ulterior credeți că vă vor ajuta la dezvoltare.*

*- Folosiți criterii pentru a grupa cerințele într-un mod care vi se pare util – după zona de tehnologie (BE, DevOps ș.a.) după eventualele module ale app (comunicare, procesare, stocarea datelor ș.a.), după feature-uri. Menționați aceste categorii și în documentul de analiză.*

Am facut clasificarea in dreptul fiecarei cerinte (cu " ->" )

*3. La acest moment puteți să creați un repo de GitHub pentru proiectul vostru și să puneți cerințele într-un back-log de issue-uri în GitHub Issues. Folosiți grupările de la 2 pentru a crea label-uri relevante pentru fiecare issue.*

*Resources:*
*The Product Backlog: A Step-by-step Guide În general sunt bune articolele de la toptal pe software engineering.*

*4. Play planning poker. Planning poker e un joc prin care toți membrii echipei evaluează prioritatea și dificultatea taskurilor în raport cu abilitățile individuale. Media acestor evaluări generează nivelul de prioritate și dificultate final al cerinței.*

*Recomand aplicația de aici: scrumpoker.online (pare că are și o integrare cu GitHub, dar nu am testat-o)*

*Pentru fiecare issue, veți juca câte două runde de planning poker. Trebuie să fiți într-un call. După ce alegeți issue-ul, fiecare user se autentifică în aplicație, și simultan ar trebui să votați dificultatea acelui issue. Faceți media și notați-o. Discutați dacă considerați că este cazul, și feel free să schimbați rezultatul dacă vi se pare că nu e potrivit.*
*Repetați pentru a identifica prioritatea taskului (adică cât de valoros este pentru aplicația finală). Notați și acest rezultat la fiecare task.*

*Dificultate*

1. *13.33*
2. *8*
3. *3.33*
4. *5.33*

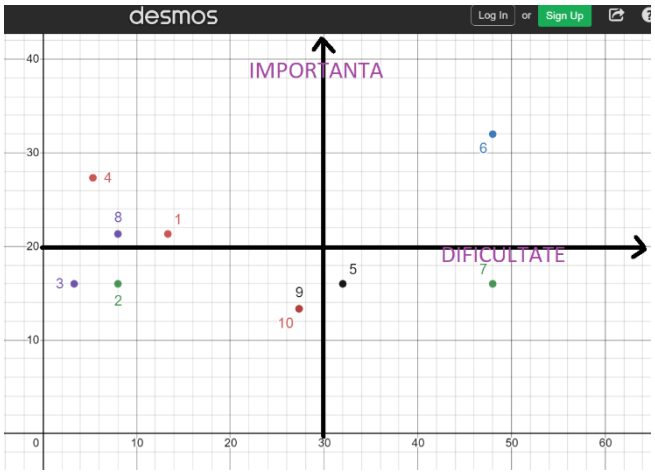
5. *32*
6. *48*
7. *48*
8. *8*
9. *27.33*
10. *27.33*

*Importanta*

1. 21.33
2. *16*
3. *16*
4. *27,33*
5. *16*
6. *32*
7. *16*
8. *21.33*
9. *13.33*
10. *13.33*

*5. Plot the issues.*

*Realizați o axă, unde pe una dintre axe aveți dificultataea, iar pe cealaltă, prioritatea. Împărțiți axa în 4 cadrane (usor-valoros, dificil-valoros, usor-nevaloros, dificil-nevaloros). Astfel toate cerințele vor fi grupate în 4 categorii. Veți prioritiza cerințele usor-valoros, veți pune în backlog cele usor-nevaloroase și fidicil-valoroase, și în nice-to-have cele dificil nevaloros.*




*6. Check the features.*

*Verificați că adunate toate issue-urile prioritare, și din backlog, adunat constituie cele 5 feature-uri minim necesare pentru a îndeplini proiectul.*

*7. Add technical issues.*

*Adăugați issue-uri doar în GitHub de care va trebui să vă ocupați care nu reies din cerințe (dev setup, deployment setup, etc.)*

## Alocarea rolurilor

*Fie alocați pentru fiecare cerință unul dintre membrii echipei care va realiza acea cerință, fie asociați membrii cu anumite label-uri, urmând ca respectivii membrii ai echipei să realizeze taskuri din labelul asociat.*

## Documentul de analiză va fi adăugat în GitHub-ul proiectului.

# Concluzii

### Cerinte implementate:

*Optiunea de a configura temperatura*

*Optiunea de a uda plantele*

*Optiunea de a configura umiditatea solului*

*Optiunea de a lua informatii despre vreme*

***Cerinte implementate partial:***

*Optiunea de a configura intervalul de timp pentru udarea plantelor*

***Cerinte care au ramas neimplamentate:***

*Configurarea luminii sistemului*

Alerte pentru cantitatea de apa