Design- och utvecklingsplan

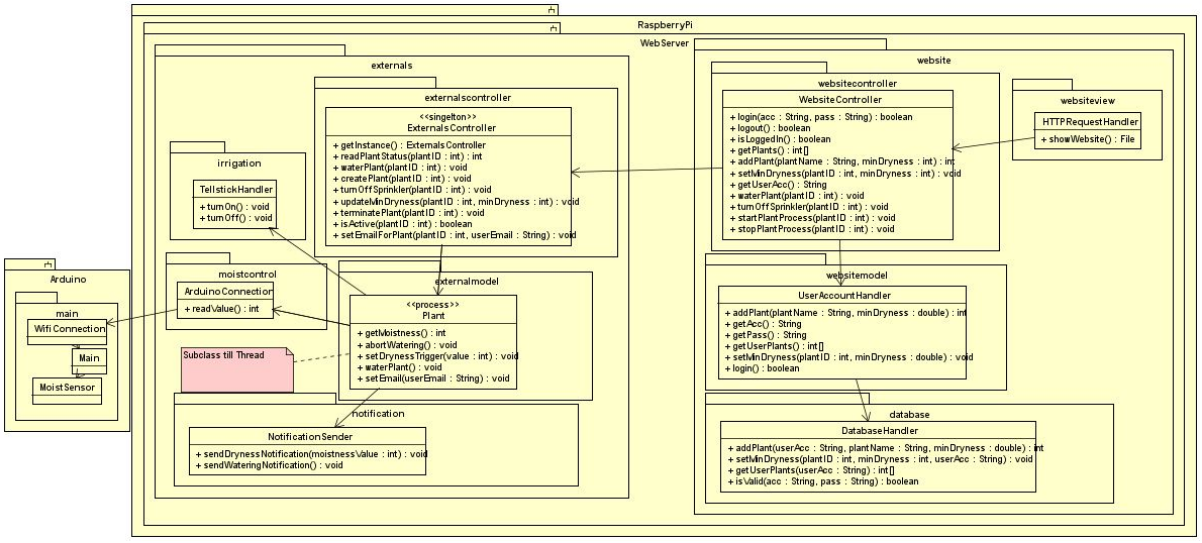
Grupp 10 - FlowerPower
Hampus Pukitis Furhoff
Senast ändrad 2018-05-21

Innehåll

Syfte

Syftet med detta dokument är att sammanfatta hur vi jobbar med koden i vårt projekt så att alla i projektgruppen jobbar på samma sätt och så att projektmedlemmar kan kolla upp saker dom är osäkra på.

Design/modell



Språk och framework

Arduino

På vår arduino har vi kommit fram till att vi ska koda i arduinos egna miljö med arduino-språket eftersom det finns mycket stöd för det och det är lätt att komma igång med. Vi övervägde att koda i C på arduinon men kom fram till att det mest skulle krångla till saker

och att dom hasighetsförbättringar man eventuellt skulle få inte var så viktiga för oss i nuläget.

Raspberry Pi

På raspberry pi'en övervägde vi att koda vår webserver i java eftersom alla i gruppen är bekanta med det men valde istället python eftersom det ska vara väldigt lätt att lära sig och likt java. Andra skäl till varför vi valde python istället för java var för att det ska finnas väldigt mycket stöd och dokumentation just om att göra saker på en raspberry pi med python.

Frameworks

Vi kör med Flask på serversidan, detta eftersom det är lätt att lära sig, lättviktigt och det går bra att göra MVC-struktur på med det. Vi kör inte med något framework på klientsidan eftersom vi inte har så stort behov av det.

Kodstandard

Variabler och funktioner skrivs med kamelnotation med inledande liten bokstav (camelCase). Klassnamn skrivs med kamelnotation med inledande stor bokstav (CamelCase). Varje funktion bör ha förklarande kommentarer men man bör undvika kommentarer inne i funktionen, skriv istället förklarande kod och undvik hårdkodade värden. Här skriver jag bara väldigt kort så om man vill gå in mer i detalj så kan man läsa “A First Course in Object Oriented Development A Hands-On Approach” av Leif Lindbäck.

Smelly code

Smelly code gör det svårare att förstå koden och kan innebära mycket onödigt arbete. För att förhindra smelly code så ska vi refaktorera när vi stöter på nedanstående företeelser.

Duplicated code

Om man har samma kod på flera ställen så är det ett problem eftersom om man behöver ändra i den koden så måste man ändra på flera ställen. Bättre då att refaktorera och lägga koden i en egen metod.

Long method

En metod är för lång om man som användare av metoden inte kan utläsa vad metoden gör bara på metodnamnet. Ett bra sätt att veta om en metod är för lång är om man som programmerare behöver lägga in kommentarer i metoden för att förklara vad metoden gör. Då är det bättre att bryta upp metoden i olika metoder.

Large class

På samma sätt som att en metod kan vara för lång så kan en klass vara för lång. Om en klass gör annat än det som utlovats av klassnamnet så bör man se över klassen.

Meaningless names

Om man inte använder namn som förklarar vad en variabel, metod eller klass är så blir koden snabbt mycket mer svårförstådd. Istället för att skriva tex pp så kan man skriva plantProcess och man förstår direkt vad det är för något.

Unnamed values

Om man använder sig av primitiva data så bör man lägga dessa i en variabel med ett förklarande namn, tex så bör man skriva plantID = 3; getPlant(plantID); istället för getPlant(3); eftersom det blir mycket tydligare vad man menar.

Versionshantering

Vi använder github för att versionshantera vår kod. Vår repository finns på https://github.com/fongie/projectgaming. Varje gång man ska jobba med en ny grej så gör man en ny branch med ett förklarande namn i camelcase. När man är klar så mergar man med test/huvud-branchen eller med en annan branch som också är klar där den nya featuren testas och till slut när alla brancher är klara och testade tillsammans mergas huvud/test-branchen med master. Mergningen till master görs enbart med godkännande av utvecklingsansvarig. Om testning visar att vissa saker behöver fixas till så görs detta i en egen branch och sedan mergar man igen till test/huvud-branchen. Som sista steg så tas arbetsbranchen bort. I master ska vi endast ha fungerande och testad kod, så därför mergar vi endast till master från test/huvud-branchen.