SYSTEM WSPOMAGAJĄCY ZDALNE ZARZĄDZANIE UPRAWĄ ROŚLIN

Autorzy: Marcin Cieślak, Damian Cywiński, Tomasz Dylak, Damian Strycharczuk

Wprowadzenie

Zamysłem projektu jest stworzenie systemu, umożliwiającego osobom fizycznym kontrolowanie warunków panujących na podlegających im uprawach rolnych, za pomocą intuicyjnego interfejsu graficznego, z poziomu smartphone'a lub aplikacji przeglądarkowej. Dzięki ciągłemu dostępowi do aktualnych danych pomiarowych, osoby zarządające uprawą będą w stanie szybko i efektywnie reagować na zmieniające się parametry, takie jak: * wilgotność powietrza * nawodnienie gleby * temperatura

Gromadzenie pomiarów z uprawy umożliwi stworzenie przydatnych wykresów oraz prognóz, pozwoli również na zautomatyzowanie niektórych czynności (np. uruchomienie zraszaczy, gdy wilgotność gleby spadnie poniżej określonego poziomu). W sytuacji, w której pomiary wskazywać będą na niekorzystny stan uprawy, system generował będzie automatyczne powiadomienia, skierowane do wyznaczonych osób, dzięki czemu będą one mogły podjąć odpowiednie czynności tylko w momencie, w którym jest to faktycznie potrzebne - ograniczony zostanie wysiłek, który w innym wypadku musiałby zostać włożony w manualne dokonywanie pomiarów oraz analizę.

Kluczowym zadaniem systemu będzie więc pomoc w dbaniu o dwa bardzo istotne czynniki * oszczędność czasu potrzebnego na zarządzanie * jakość uprawy (jej obfitość, szybki wzrost, zdrowie)

Dzięki temu zaoszczędzone zasoby mogą zostać przeniesione na nowe uprawy, bez konieczności martwienia się o spadek ogólnej jakości.

Wymagania funkcjonalne

Użytkownik, za pośrednictwem interfejsu graficznego może:

- dodawać / usuwać uprawy
- zmieniać nazwę oraz kolor znaczący uprawy (kolor pomoże mu odróżnić wizualnie jedną uprawę od drugiej)
- przeglądać historię zmian kluczowych dla uprawy parametrów (wilgotność powietrza, nawodnienie gleby, temperatura) w czasie
- kontrolować aktualne kluczowe dla uprawy parametry
- ustalać wskaźniki alarmowe dla każdego z parametrów
- ustalać sposób powiadomień na wypadek przekroczenia wskaźników
- deklarować czynności zautomatyzowane przy pomocy prostej składni bądź formularza (np. uruchamianie zraszaczy w momencie osiągnięcia poziomu X przez parametr A)

Użytkownik otrzyma dostęp do tych możliwości dopiero po rejestracji oraz logowaniu. Wówczas zostanie stworzona dla niego prywatna przestrzeń robocza, za pośrednictwem której będzie mógł zarządać swoimi uprawami.

Każda nowa uprawa wymagać będzie zainstalowania Arduino oraz modułu Arduino w fizycznej lokalizacji uprawy oraz dostosowania go do współpracy z serwerem * konfiguracja sieci * uruchomienie modułów, konfiguracja (adresy IP, hasła, kluczowe parametry)

Wymagania niefunkcjonalne

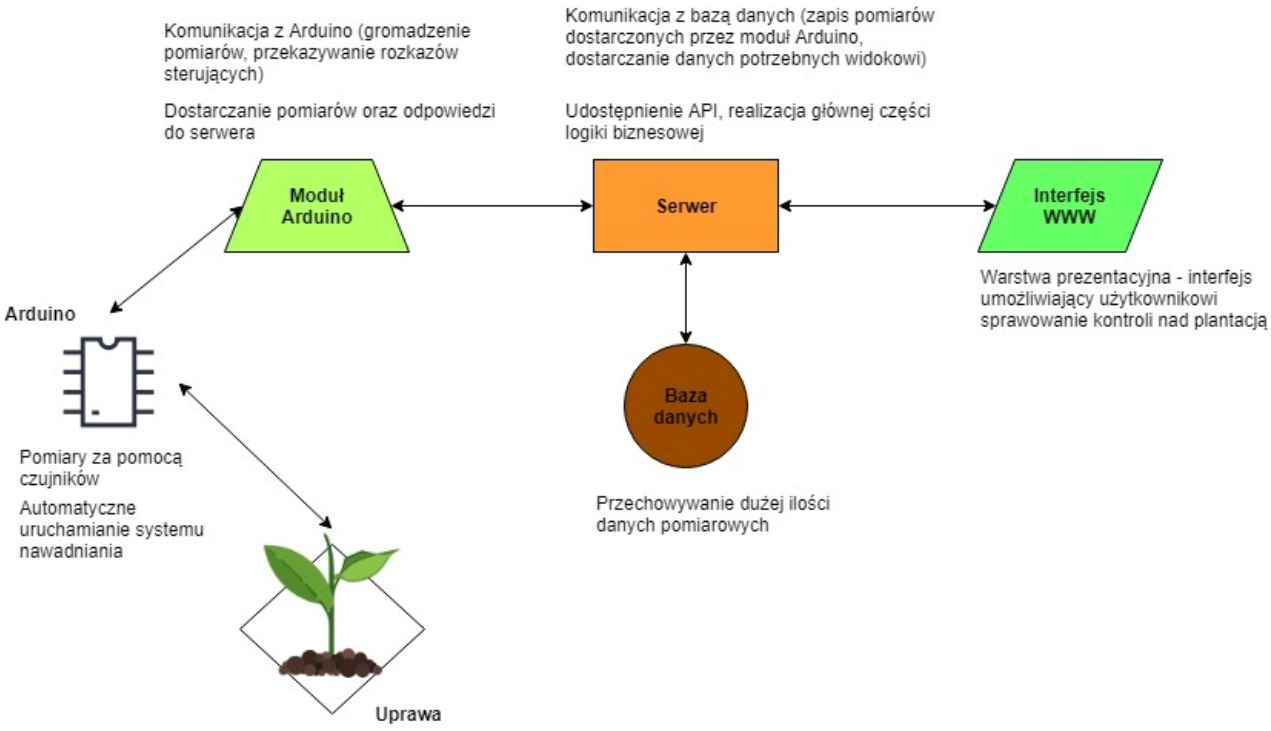
- Bezpieczeństwo systemu - odporność na podstawowe ataki (Sql Injection, XSS), zabezpieczenie komunikacji pomiędzy modułami (SSL), przechowywanie zakodowanych haseł w bazie (SHA-2), testy
- Intuicyjny interfejs - prosta obsługa dla osób nietechnicznych, czytelna czcionka, ograniczenie elementów do minimum, kluczowa funkcjonalność systemu zawsze pod ręką, ograniczenie grafik - skorzystanie z jednoznacznych ikon, responsywny design
- Wysoka dostępność - odporność na awarie, modułowa architektura umożliwia wprowadzenie nadmiarowości (alternatywnych ścieżek na wypadek awarii)
- Skalowalność - system powinien być gotów na łatwą rozbudowę w przyszłości (wykorzystanie wzorców projektowych, framework'ów, zachowanie czystości kodu)
- Aktualna (stale rozwijana) dokumentacja dla użytkowników oraz dokumentacja techniczna
- Ograniczenie danych trafiających do bazy (np. pomiary co określony odstęp czasu), zapewnienie odpowiedniej przestrzeni dyskowej pod dużą ilość danych, gotowość warstwy bazodanowej na duży rozrost
- Poznanie oraz zrozumienie fachowości związanej z realną uprawą roślin

Ryzyka

- Nieznajomość oraz konieczność uczenia się nowych technologi przez zespół - to co na początku wydaje się łatwe, z czasem może okazać się trudne w implementacji
- Spadek motywacji oraz rozbieżne motywacje wśród członków zespołu
- Czynniki zewnętrzne wpływające na dostępność członków zespołu (choroba, sytuacja rodzinna)
- Niejasne, trudne lub zmieniające się wymagania projektowe oraz brak jasnego harmonogramu (to ryzyko musi zostać zniwelowane już na samym początku!)
- Nieznajomość dokładnej sytuacji rynkowej, zapotrzebowań istniejących wśród osób faktycznych, dbających o istniejące uprawy rolne (ryzyko może zostać zniwelowane dzięki wywiadowi społecznemu)
- Ograniczone zasoby finansowe oraz czasowe możliwe do przeznaczenia na projekt (presja czasu może wpłynąć na jakość kodu, mniejsze pokrycie testami, etc.)

Architektura systemu

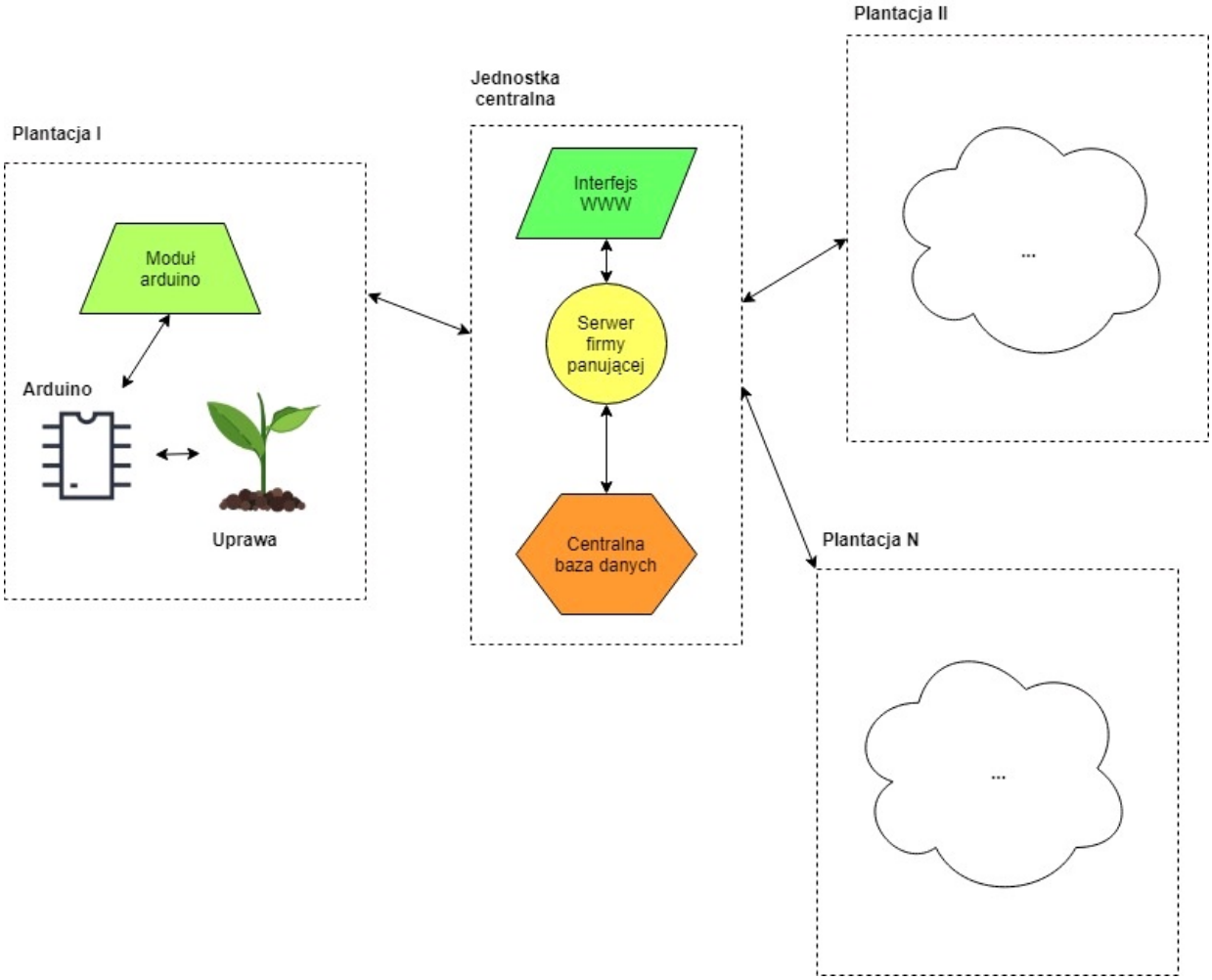
Podejście ogólne



Model z centralnym serwerem

Zalety * uprawy nie są odpowiedzialne za serwer, bazę danych ani interfejs WWW - jest on zapewniany centralnie * mniejsze ryzyko błędnej konfiguracji * łatwiejsze powołanie systemu do życia

Wady * zapewnienie zasobów pamięciowych oraz mocy obliczeniowej do obsługi wszystkich upraw leży po stronie centralnej administracji * na wypadek awarii jednostki centralnej, wszystkie uprawy przestają działać



Model "samowystraczalny"

Model samowystarczalny to pod względem schematu po prostu Podejście ogólne, powielone na każdą uprawę. W takim modelu każda uprawa posiada własny serwer, bazę danych oraz interfejs.

Zalety * awaria jednej uprawy nie wpływa na pozostałe * elastyczne rozdzielenie zasobów pamięciowych oraz mocy obliczeniowej - mniejsza ilość danych (dane dotyczące tylko konkretnej uprawy)

Wady * odpowiednia infrastruktura sieciowa oraz sprzęt muszą zostać zapewnione na miejscu (uprawie) * powołana musi zostać dodatkowa jednostka techniczna (administracyjna) odpowiedzialna za każdą uprawę * trudniejszy proces instalacji oraz konfiguracji

Pyhton, Django

Język Python został stworzony we wczesnych latach 90-tych (1991 roku) przez holenderskiego programistę Guido van Rossum. Istotny wkład w rozwój języka pochodzi od społeczności Python'a, która z roku na rok przybiera większy zakres na świecie, nadając mu nową jakość. Python jest językiem programowania starającym się zawrzeć w sobie najlepsze rozwiązania i intencje innych języków programowania, które pojawiły się i były udoskonalane w przeciągu ostatniego dwudziestolecia. Jest on interpretowanym, obiektowym, wysokopoziomowym językiem co ułatwia jego testowanie i stosowanie w sposób interaktywny. Python to oprogramowanie typu Open-Source zarządzany przez Python Software Foundation, działające na wielu platformach, takich jak: GNU/Linux, Mac OS czy Windows.

Django to Open-Source'owy framework przeznaczony do tworzenia aplikacji internetowych, napisany w Pythonie. Powstał pod koniec 2003 roku jako ewolucyjne rozwinięcie aplikacji internetowych, tworzonych przez grupę programistów związanych z Lawrence Journal-World. W 2005 roku kod Django został wydany na licencji BSD. Django realizuje wzorzec architektoniczny model-template-view (pokrewny z MVC). Cechują go m.in:

- Automatycznie generowany i kompletny panel administracyjny, z możliwością dalszego dostosowywania
- Przyjazne adresy dokumentów z możliwością dowolnego ich kształtowania
- Prosty lecz funkcjonalny system szablonów czytelny zarówno dla grafików jak i dla programistów
- Oddzielenie logiki aplikacji (widok), logiki biznesowej (model), wyglądu (szablony) oraz baz danych
- Wsparcie dla wielojęzycznych aplikacji
- Bardzo duża skalowalność i wydajność pod obciążeniem
- Wydajne systemy cache'owania, obsługa Memcached
- Własny, prosty serwer do testowania aplikacji

Front-End

Założenia:

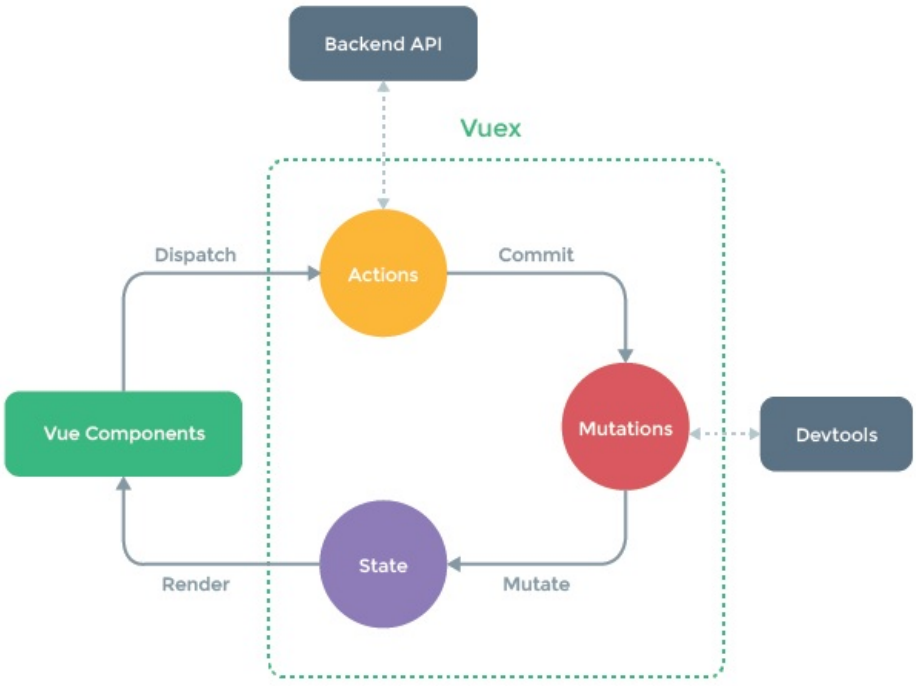
- prosty oraz intuicyjny interfejs
- dynamiczne odświeżanie elementów strony (bez konieczności ręcznego odświeżania - Ajax)
- czytelne czcionki, brak elementów rozpraszających użytkownika
- wykorzystanie prostych ikon, ułatwiających odnalezienie konkretnych elementów na stronie
- ograniczenie ilości zapytań kierowanych do serwera
- dobre ustrukturozywanie kodu (zachowanie zasad Clean Code, zapewnienie łatwej rozszerzalności)
- łatwy dostęp do wszystkich elementów interfejsu (głębokość na maksymalnie trzy kliknięcia)
- spójne formatowanie kodu (zapewnione dzięki IDE - Visual Code)
- podział elementów składowych systemu na małe komponenty (zarządzanie mniejszymi fragmentami jest o wiele łatwiejsze)
- (możliwa obsługa wielu języków)

Vue.js

Wybrany został framework Vue.js pozwalający tworzyć dynamiczne strony WWW przy wykorzystaniu javascript. Vue jest bardzo dobrze udokumentowany oraz zawiera szereg pomocnych bibliotek. Pozwala w bardzo prosty sposób podzielić widok na mniejsze komponenty, które następnie mogą być używane w wielu miejscach systemu. Ponadto praca nad mniejszymi fragmentami zawsze jest prostsza.

Vuex

Vuex jest rozszerzeniem Vue.js, funkcjonującym jako centralny magazyn gromadzący dane, z których następnie (w kontrolowany sposób) mogą korzystać wszystkie komponenty widoku.



Jest to bardzo przydatne rozszerzenie, zwłaszcza biorąc pod uwagę to, że klient pobierał będzie od serwera dane, wykorzystywane następnie w wielu miejscach. Klient będzie również zmieniał te dane. W związku z tym nie tylko serwer będzie musiał zostać powiadomiony o zmianie, ale i każdy komponent widoku. Dzięki Vuex utrzymanie spójności jest bardzo proste.

Axios

Axios pozwala w bardzo prosty sposób tworzyć zapytania do API (serwera)

```
const axios = require('axios');

// Make a request for a user with a given ID
axios.get('/user?ID=12345')
```

```
.then(function (response) {
  // handle success
  console.log(response);
})
.catch(function (error) {
  // handle error
  console.log(error);
})
.then(function () {
  // always executed
});
```

W celu umożliwienia łatwej zmiany biblioteki, stworzone zostaną specjalne service, wykorzystywane przez klienta do komunikacji z serwerem. Dzięki temu ewentualna zmiana biblioteki służącej do komunikacji z serwerem będzie łatwa - wystarczy wprowadzić nową implementację serwisu.

i18n

Rozszerzenie umożliwiające wprowadzenie wielu wersji językowych.

Bootstrap

Bootstrap jest frameworkiem, który pozwala na tworzenie eleganckich, responsywnych widoków, przy użyciu HTML oraz CSS. Posiada szereg zdefiniowanych klas, dzięki czemu tworzenie widoku jest szybkie.

Autoryzacja przy pomocy tokena JWT

Jako że w komunikacji klient-serwer stawiamy na podejście REST, token JWT będzie idealnie pasował do naszego systemu. Klient, w momencie logowania, otrzyma od serwera specjalny token, w którym zakodowana jest tożsamość logującego się użytkownika. Serwer, dzięki mechanizmom wewnętrznym, potrafi ocenić prawdziwość tokena oraz pobrać z niego obiekt użytkownika, który się nim posługuje. Dzięki temu zabezpieczone zostaną zasoby, do których dostęp powinien mieć tylko zalogowany użytkownik oraz zwrócone zostaną tylko i wyłącznie zasoby należącego do konkretnego użytkownika.

Repozytorium Git

GIT to system kontroli wersji na licencji Open-source, który pozwala zapamiętać i synchronizować pomiędzy użytkownikami zmiany dokonywane na plikach. Umożliwia przywołanie dowolnej wcześniejszej wersji, a co najważniejsze, automatycznie łączy zmiany, które ze sobą nie kolidują, np. dokonane w różnych miejscach w pliku.

Trello

Trello jest narzędziem pozwalającym na skuteczne zarządzanie projektami stworzonym przez y Fog Creek Software w 2011 roku. Prawa do programu aktualnie znajdują się w rękach Atlassian.

Baza danych

MariaDB to system zarządzania relacyjnymi bazami danych stworzona przez grupę (głównie) byłych pracowników MySQL AB, pod przewodnictwem Michaela Wideniusa, współtwórcy MySQL. Celem głównym projektu jest współpraca ze społecznością wolnego oprogramowania i udostępnianie jej na licencji GPL, w przeciwieństwie do niepewnego statusu licencji MySQL, która zależy teraz od firmy Oracle. System ten jest kompatybilny z MySQL. Jest on również lepiej zoptymalizowany od MySQL, co przekłada się na zwiększoną wydajność.

Motywacje technologiczne:

- dobra dokumentacja
- rozwinięta społeczność = szybka pomoc
- frameworki = konkretne wymagania dot. struktury projektu, prowadzenia kodu = łatwiejsza rozbudowa w przyszłości, łatwiejsza współpraca
- Python = Django umożliwia szybkie oraz wygodne tworzenie back-endu

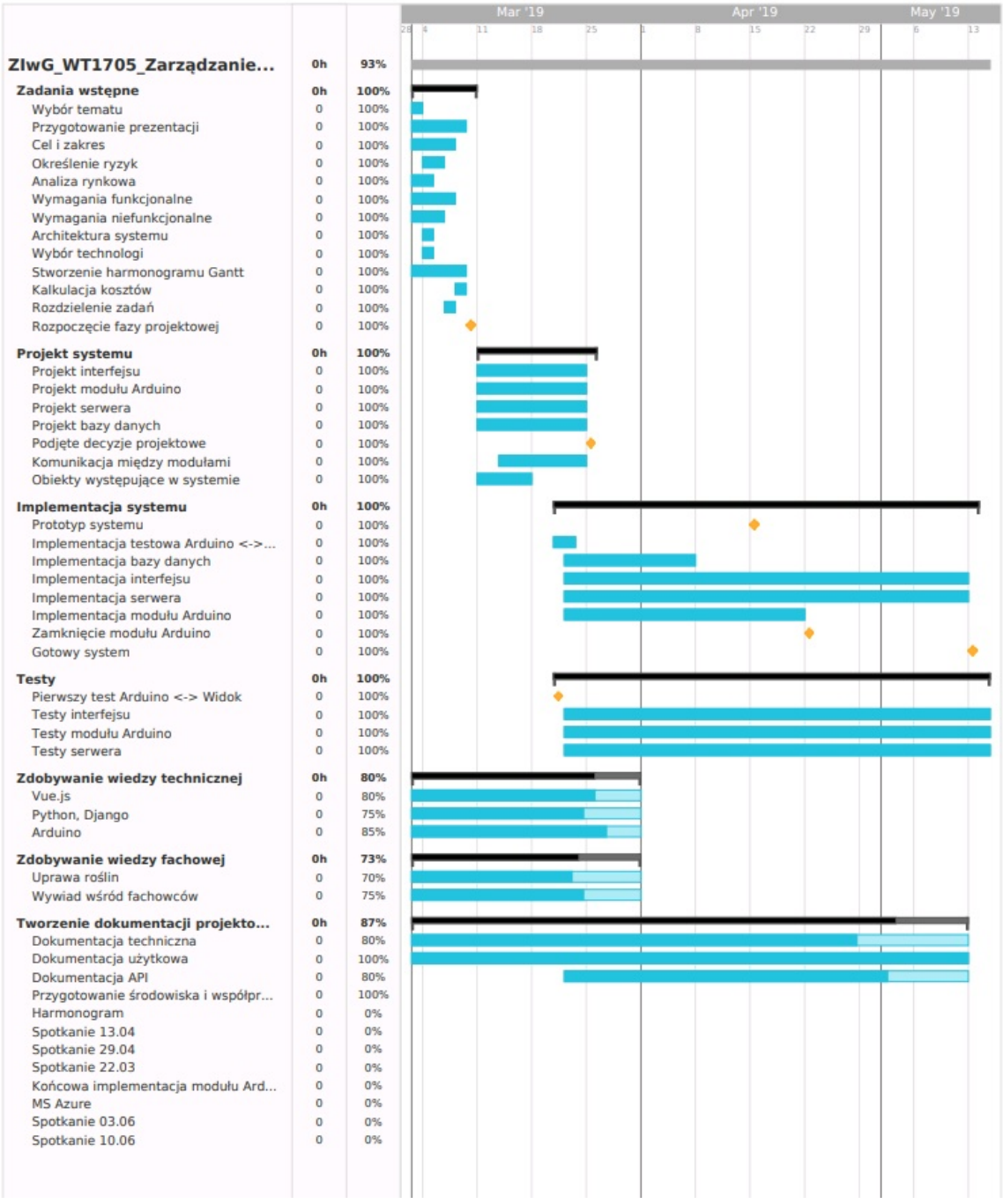
Aktualny stan rynku

Analiza rynku jest trudna, ze względu na niewielką konkurencję oraz brak możliwości przetestowania innych aplikacji. W celu usprawnienia aplikacji konieczny będzie wywiad wśród potencjalnych użytkowników.

Potencjalna, znana konkurencja:

Harmonogram oraz nakład czasowy

Wykres Gantt'a



Nakład czasowy

Nakład czasowy to około:

```
90 h x 4 osoby= 360h pracy w domu
```