

Dokumentacja Illumination

Główny opis projektu	3
Motywacja	3
Cel projektu	3
Przedział czasowy	5
Spotkania	5
Daily	5
Planning	5
Sprint review	5
Retrospective	5
Backlog refinement	5
Członkowie projektu	6
Wykonana lista zadań dla członków	7
Milestones	8
I semestr	8
I milestone - 01.11.2020	8
II milestone - 02.01.2020	8
III milestone - 17.02.2021 (koniec semestru I)	8
II Semestr	8
I milestone - 01.02.2020	8
II milestone - 10.03.2020	8
III milestone - 01.05.2021 (koniec semestru II)	8
Epiki	9
M-1 Dokumentacja	9
M-1 Jira	9
M-2 UML	9
M-3 Test Scenarios	9
M-3 Github	9
Opis i motywacja dla wyboru technologii	10
Django	10
React	10
Electron	10
Python	10
Jenkins	10
Github	10
Docker	10
Dokumentacja Techniczna	11
Use Case Diagram	11

Class Diagram	11
Sequence Diagram	11
Activity Diagram	11
Zrzuty ekranu GUI z opisem workflow	11
Infrastruktura sprzętowa	12
Środowiska	12
Dev (unstable)	12
Staging (stable)	12
Prod (stable)	12
Software versioning - Releases	13
Schemat numerowania wersji	13
Pre-alpha	13
Alpha	13
Beta	14
Release candidate	14
Stable release	14
Priorytetyzacja	15
Wycena Punktów	15

Główny opis projektu

Budynek Instytutu Informatyki Uniwersytetu Marii Curie-Skłodowskiej został wyposażony w oświetlenie w każdym z szklanych okien. Każde okno posiada w sobie panel led-ów RGB, który można zaprogramować.

Aplikacja będzie pozwalać użytkownikom na tworzenie scen i animacji, które będą wyświetlane na Budynku Instytutu. Dodatkowo moderator będzie posiadał możliwość zarządzania wyświetlaną animacją i kolejką przyszłych animacji.

Motywacja

Motywacją do stworzenia tego projektu jest promowanie Instytutu UMCS. Wierzymy, że możliwości narzędzia do tworzenia animacji w prosty sposób, które następnie można wyświetlić na budynku pozwoli zwiększyć zainteresowanie kierunkiem Informatyka w kolejnych latach.

Cel projektu

Celem projektu jest umożliwienie końcowym użytkownikom na łatwą i szybką kontrolę wyświetlanych konfiguracji na budynku, zarządzanie kolejką oraz tworzenie i emulowanie nowych animacji.



Uniwersytet Marii Curie-Skłodowskiej... 53m



auseika_



@umcs_lublin

Przedział czasowy

Początek - 05-10-2020

Koniec - 02-03-2021

Spotkania

Długość sprintów to **2 tygodnie**.

Daily

Co drugi dzień o 18:00

Planning

Poniedziałek 18:00

Sprint review

Niedziela 18:00

Retrospective

Niedziela 18:30

Backlog refinement

Niedziela 17:00

Członkowie projektu

Nasza motywacja oraz zobowiązania każdego z członków znajduje w zdjęciu poniżej.

<https://illumination.atlassian.net/wiki/spaces/IL/pages/14581793/Project+roles>

Project roles

Leader - @Maciej Prostek

Competences - Most knowledge about the project. Knows what features are needed and the time it takes to implement them.

- Add issues, supervise
- Plan sprints with realistic dates
- Assign backlog tasks to users
- Present the project (**final**)

Engineer - @Mateusz Moru

Competences - Worked on backend apps. Knows how to work well with databases.

- Implement features on backend (**Django**)
- Fix bugs found by tester
- Develop and test on branch **development**

Engineer / Devops - @Szymon Szostak

Competences - Most knowledge about setting up hardware and Docker. Worked in robotics. Knows how to configure raspberry pi.

- Implement features on queue (**Python/Node.js**)
- Configure Docker and Docker-Compose
- Develop and test on branch **development**
- Configure environment **DEV, TEST, STAGE, PROD**
























Tester - @Dorota Julia

Competences - Thinks out of the box when using software. Knows how to find corner cases. Really good at algorithms.

- Focus on end-to-end tests
- Write tests using pytest for **Queue** and **Django**
- Use <https://en.wikipedia.org/wiki/Given-When-Then> while writing tests
- Test on branch **release**
- Create test scenarios

1. Maciej Prostek - **Lider**
2. Dorota Zaremba - **Tester**
3. Mateusz Moruś - **Inżynier**
4. Szymon - **Inżynier/Devops**

Wykonana lista zadań dla członków

Completed issues						View in issue navigator
Key	Summary	Issue type	Epic	Status	Assignee	Issue count
IL-3	Create diagram use-case	 Documentation		 DONE		 1
IL-4	Create pre-documentation	 Documentation		 DONE		 1
IL-13	Główny opis projektu	 Documentation		 DONE		 1
IL-14	Motywacja i cel projektu	 Documentation		 DONE		 1
IL-15	Opis milestones	 Documentation		 DONE		 1
IL-16	Motywacja wyboru technologii	 Documentation		 DONE		 1

Milestones

Przez cały projekt będą **trzy** milestones. Każdy z nich będzie trwać **2 miesiące**.

I semestr

I milestone - 01.11.2020

Podłączenie Jiry, wstępny zarys dokumentacji. Podzielone zadania do dokumentacji.

II milestone - 02.01.2021

Diagramy UML.

III milestone - 17.02.2021 (koniec semestru I)

Dokończenie dokumentacji oraz opisanie testów.

II Semestr

I milestone - 01.02.2021

Przygotowanie wszystkich środowisk. Implementacja kolejki.

II milestone - 10.03.2021

Produkt MVP. Działające CI/CD na Jenkins.

III milestone - 01.05.2021 (koniec semestru II)

Dokończona implementacja produktu. Kod będzie open-source'owy, otwarty na kolejne zmiany i iteracje dla przyszłych członków koła naukowego.

Epiki

M-1 Dokumentacja

- Czas trwania: 04.10.2020 - 03.11.2020
- Dokumentacja
- Stworzenie ogólnego planu projektu

M-1 Jira

- Czas trwania: 04.11.2020 - 03.12.2020
- Dodanie członków, rozpisanie tasków

M-2 UML

- Czas trwania: 04.12.2020 - 03.01.2021
- Tworzenie diagramów UML

M-3 Test Scenarios

- Czas trwania: 04.01.2021 - 04.02.2021
- Dodanie opisy testów przygotowane pod implementację

M-3 Github

- Czas trwania: 04.01.2021 - 04.02.2021
- Dodanie kontroli wersji do naszego projektu

M-4 Inicjalizacja projektu

- Czas trwania: 04.01.2021 - 04.02.2021

M-4 I

Opis i motywacja dla wyboru technologii

Django

Wybór technologii back endowej był bardzo prosty. Chcieliśmy użyć **Django** z powodu panelu admina. Umożliwiło nam to szybkie administrowanie danych bez implementacji logowania. W prosty sposób możemy dodawać moderatorów do akceptowania animacji oraz zarządzanie nim.

React

Wybraliśmy **React** jako technologię front endową z powodu łatwego zarządzania większą ilością komponentów. Dane często są zmieniane i komponenty mogą dynamicznie się zmieniać.

Electron

Wybraliśmy **electrona** ponieważ łatwo można podpiąć się do strony napisanej w React. Komunikacja IPC umożliwia nam sprawną komunikację między natywnym softwarem i stroną. Jest również wieloplatformowy, umożliwiający pisanie nowych animacji w każdym środowisku.

Python

Do kolejki wybraliśmy **Python**. Za jego pomocą będziemy obsługiwać zdarzenia takie jak dodawanie czy losowanie animacji. Łatwo połączymy się z urządzeniem zewnętrznym do wysyłania konfiguracji, czy również połączymy się z bazą danych.

Jenkins

Wybraliśmy do CI/CD **jenkins** bo jest łatwe w użyciu. Ma bardzo dużą społeczność użytkowników i jest open-source. Jest bardzo elastyczne i można go skonfigurować pod każdy projekt. Dużo firm go używa więc umiejętność posługiwania się tym narzędziem może być przydatne w przyszłości.

Github

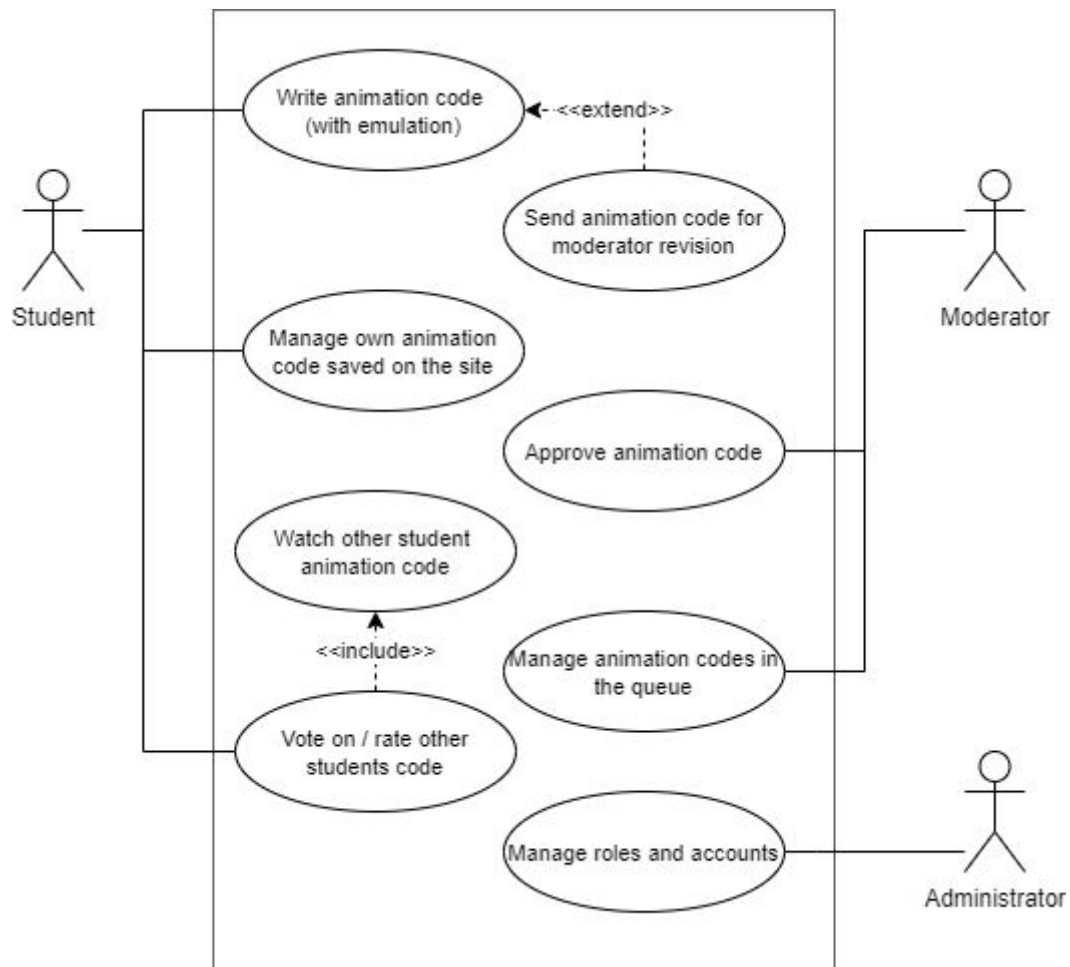
Do kontroli wersji wybraliśmy **Github** z powodu, że jest to największy serwis jeśli chodzi o otwartoźródłowy kod. Projekt jest zamieszczony w naszej organizacji kołowej, która umożliwi nam łatwe przekazanie projektu kolejnym osobom..

Docker

Będziemy używać narzędzia **Docker** do konteneryzacji. Za pomocą **docker-compose** będziemy mogli łatwo zarządzać naszymi środowiskami. Umożliwia nam to trzymanie się jednej wersji bazy danych, systemu czy też wersji pythona.

Dokumentacja Techniczna

1. Use Case Diagram



2. Class Diagram

https://app.diagrams.net/#G1KFZRw6hDm35jo6_c5u8Vp_c6RxT-K9O8

3. Sequence Diagram

<https://app.diagrams.net/#G1lbf3bLTaSZaDvYcdoa1yn38MAS8TaCu1>

4. Activity Diagram

<https://app.diagrams.net/#G1xydDBPpU64H7TG7sIL760IzxtIWXAjaH>

Zrzuty ekranu GUI z opisem workflow

Infrastruktura sprzętowa

1. Serwer z Ubuntu 20.04 i zainstalowanym Dockerem.
2. Raspberry Pi 3
3. Konwerter usb - RS485

Środowiska

Dev (unstable)

W tym środowisku będziemy dodawać wszystkie nowe funkcjonalności oraz poprawki. Nie zawsze będą przechodzić testy oraz pokrycie nie będzie zawsze wysokie.

Staging (stable)

Po wstępnym przetestowaniu software'u (przez inżyniera i testy jednostkowe) budujemy wszystko w tym środowisku oraz wykonujemy dodatkowe testy end-to-end. Tutaj będziemy korzystać z software'u jakby miało być już wystawione na produkcji.

Prod (stable)

Po przetestowaniu wszystkiego na staging możemy dodać nowy release na production. W razie jakichkolwiek bugów możemy zawsze zastosować hotfix w celu szybkiego naprawienia bug'a.

Software versioning - Releases

Cały codebase będzie trzymany na serwisie **Github**.

Nowe wersje wypuszczane na produkcję będą budowane jako aplikacja **.exe** do instalacji na środowisku **windows / linux / macos**.

Schemat numerowania wersji

Wykorzystujemy schemat numerowania wersji, aby śledzić różne wersje oprogramowania.

Przykład:

Tag name	Description	Packaged files
9.0.0	9.0.0 release (core)	drupal-9.0.0.tar.gz, drupal-9.0.0.zip
8.x-2.0-alpha6	8.x-2.0-alpha6 release (contrib)	myproject-8.x-2.0-alpha6.tar.gz myproject-8.x-2.0-alpha6.zip
9.0.0-beta1	9.0.0-beta1 release (core)	drupal-9.0.0-beta1.tar.gz, drupal-9.0.0-beta1.zip
8.x-2.3	8.x-2.3 release (contrib)	myproject-8.x-2.3.tar.gz, myproject-8.x-2.3.zip
1.0.0	1.0.0 release (contrib)	myproject-1.0.0.tar.gz, myproject-1.0.0.zip
1.0.0-beta1	1.0.0-beta1 release (contrib)	myproject-1.0.0-beta1.tar.gz, myproject-1.0.0-beta1.zip

Zródło: <https://www.drupal.org/docs/develop/git/git-for-drupal-project-maintainers/release-naming-conventions>

Pre-alpha

Pre-alpha odnosi się do wszystkich czynności wykonywanych podczas projektu oprogramowania przed formalnym testowaniem.

Alpha

Alfa to pierwsza faza testowania oprogramowania

Beta

Faza Beta zwykle rozpoczyna się, gdy oprogramowanie jest kompletne, ale prawdopodobnie zawiera wiele znanych lub nieznanych błędów. Oprogramowanie w fazie beta zazwyczaj zawiera o wiele więcej błędów niż ukończone oprogramowanie, problemy z szybkością lub wydajnością i może nadal powodować awarie lub utratę danych.

Release candidate

Release Candidate, jest wersją beta, która potencjalnie może być stabilnym produktem, który jest gotowy do wydania, chyba że pojawią się istotne błędy. Na tym etapie stabilizacji produktu wszystkie funkcje produktu zostały zaprojektowane, zakodowane i przetestowane przez co najmniej jeden cykl beta bez znanych błędów klasy showstopper

Stable release

Jest ostatnią wersją, która przeszła wszystkie weryfikacje/testy. Pozostałe błędy są uznawane za dopuszczalne. To wydanie trafia do produkcji.

Priorytetyzacja

Będziemy używać metody MoSCoW. W **backlog refinement** w Jirze zaznaczamy priorytety strzałkami. Na początku omówieniu sprintu wiemy na czym się najbardziej skupić i możemy wycenić punktowo każdy task.

Wycena Punktów

- 1 punkt - Łatwy task, nie zajmuje więcej niż dzień
- 2 do 3 punktów - Średni task, nie powinien zajmować cały sprint
- 5 punktów - Task będzie zajmować cały sprint
- > 5 punktów - Rozbicie na dwa taski