Wyższa Szkoła Ekonomii
i Informatyki w Krakowie

Projektowanie profesjonalnej aplikacji mobilnej lub webowej (P1)

PLANOWANIE PROJEKTU

Wykonały studentki 3 roku
studiów stacjonarnych na kierunku
Informatyka i Ekonometria
Oleksandra Stepankovska oraz
Viktoriya Melnyk

Aplikacja webowa

‘WELLY’

Twój asystent w trójboju siłowym

Cel projektu

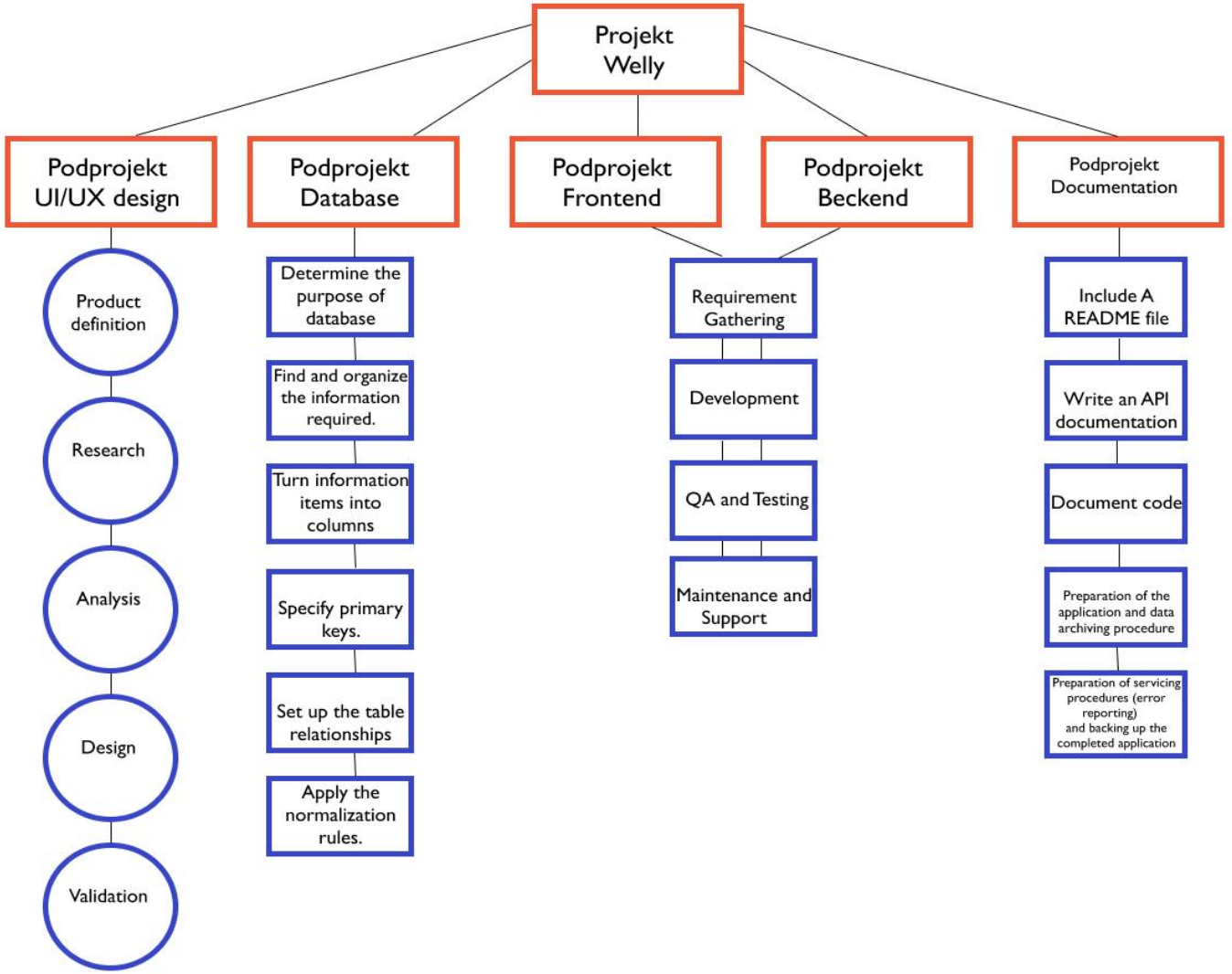
Jednym z głównych celów projektu jest **popularyzacja sportu**, jakim jest **trójbój siłowy,** co z kolei doprowadzi do pozyskania nowych inwestorów i aktywnego **rozwoju infrastruktury sportowej**.
W chwili obecnej na rynku **nie ma narzędzia**, które dostarczałoby jasnych i uporządkowanych informacji o możliwości udziału w zawodach w różnych federacjach.
Również powszechność korzystania z serwisu będzie miała wpływ na wprowadzenie nowych podejść do tych działań, gdyż polityka ich realizacji jest w tej chwili dość skomplikowana ze względu na brak dostępnych źródeł informacji. Celem projektu jest nie tylko usprawnienie aktywności dotychczasowych sportowców, ale także **zachęcenie nowych osób do udziału w zawodach**.

Aplikacja dostarcza wszystkie niezbędne informacje o uczestniczeniu, mianowicie:

- *Lista federacji*
- *Informacje o członkostwie*
- *Kalendarz zawodów*
- *Protokoły zawodowe*
- *Lista dozwolonego sprzętu w różnych federacjach*
- *Wykaz przepisów ogólnych*
- *Informacje o kontroli antydopingowej*

Inne uwagi, które są obecne w poszczególnych federacjach.

OKREŚLENIE STRUKTURY HIERARCHICZNEJ PROJEKTU



PODZIAŁ PROJEKTU

Imię	Czynności
Viktoriya Melnyk	• *UI/UX projekt* • *Frontend* • *Dokumentacja*
Oleksandra Stepankovska	• *Baza danych* • *Testowanie aplikacji* • *Backend* • *Dokumentacja*

CHARAKTERYSTYKA UŻYWANYCH NARZĘDZI / TECHNIKALIÓW

W trakcie realizacji projektu zostaną wykorzystane następujące technologie:

- Jira
- Miro
- Figma
- React
- GraphQL
- Hapi.js
- Node
- MongoDB wraz z mogoose
- nodemon
- Postman
- Apollo
- Swagger
- Fiddler
- npm
- jmeter
- Azure DevOps
- Git/Github

Figma to program w stylu Sketcha lub Adobe XD, który służy do tworzenia projektów stron internetowych oraz aplikacji mobilnych. W przeciwieństwie do Sketch i InVision Studio, Figma jest rozwiązaniem wieloplatformowym (działa na Windows, Mac i telefonach komórkowych). Działa w przeglądarce, co oznacza, że nie musisz pobierać żadnych dodatkowych programów na swój komputer.
To jest darmowy produkt. Cała praca w Figma odbywa się w chmurze. Figma udostępnia projekt nie tylko projektantom, ale także wszystkim pozostałym członkom zespołu.

Stworzymy bardzo elastyczne i wydajne API oparte na Node.js i GraphQL z dokumentacją stworzoną na Swagger. Podstawą API będzie Hapi.js - mało znany framework, który pozwala tworzyć logikę programu bez rozpraszania się na architekturę, a także zyskać większą kontrolę nad procesem niż może zaoferować np. Restify lub Express.

Nasz projekt zakłada tworzenia interfejsów użytkownika poprzez React.

Użyjemy GraphQL do zapytań.
GraphQL to otwarty język zapytań i manipulacji danymi dla API i środowiska wykonawczego do obsługi zapytań z dostępnych danych.
GraphQL pomaga uporać się z wieloma nieprzyjemnymi momentami, które mogą trafić się tradycyjnym API REST. Tutaj jest kilka z nich:
Over-fetching - odpowiedź zawiera dane, które nie są używane.
Niepełne pobieranie — niewystarczające dane uzyskane po próbkowaniu, co prowadzi do ponownego żądania.
Ogólnie rzecz biorąc, ta technologia pomoże nam uzyskać bardziej ustrukturyzowane dane.

Apollo jest łącznikiem pomiędzy naszym serwerem Hapi a GraphQL.
Dla API będzie stworzona dokumentacja. W tym celu skorzystamy z potężnego narzędzia Swagger, które pozwala w pełni wykorzystać specyfikację OpenAPI.

Nodemon - chodzi o to, aby podczas tworzenia oprogramowania nodemon śledził zmieniane przez nas pliki i po prostu restartował serwer, jeśli te pliki należą do serwera po stronie kodu.

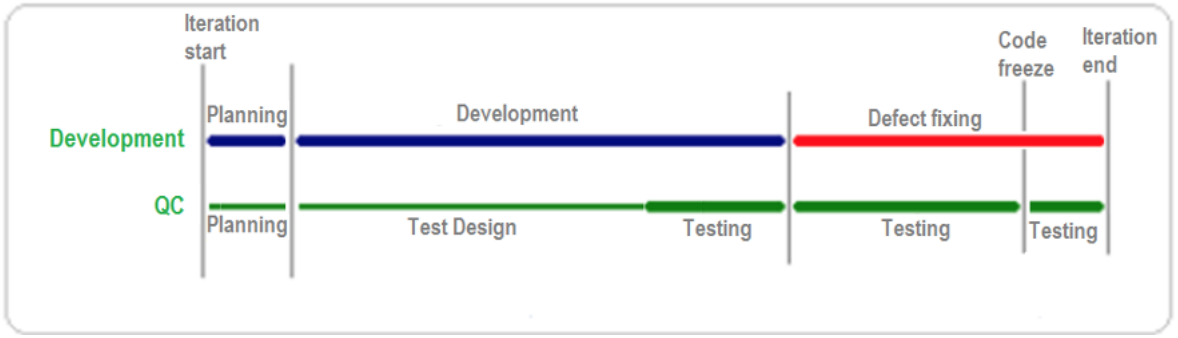
React to biblioteka JavaScript do pracy z interfejsami (UI), stworzona przez programistów Facebooka. Biblioteka była używana na stronie tego portalu społecznościowego w 2011 roku. A w 2013 roku Facebook ujawnił kod źródłowy Reacta. Dzięki React programiści tworzą aplikacje internetowe, które zmieniają sposób wyświetlania bez przeładowywania strony. W rezultacie programy szybko reagują na działania użytkownika, takie jak wypełnianie formularzy, stosowanie filtrów, dodawanie elementów do koszyka i tak dalej.
React służy do wyświetlania komponentów interfejsu użytkownika. Biblioteka może również w pełni zarządzać frontendem. W tym przypadku React używa bibliotek do zarządzania statusem i routingiem, takich jak Redux i React Router.

Powody, dla których zespół wybrał React:
- Wirtualny DOM może zwiększyć wydajność aplikacji o dużym obciążeniu, co może zmniejszyć prawdopodobieństwo ewentualnych niedogodności i poprawić wrażenia użytkownika;
- Zastosowanie podejścia izomorficznego pomaga szybciej renderować strony, dzięki czemu użytkownicy czują się bardziej komfortowo podczas pracy z aplikacją. Wyszukiwarki lepiej indeksują takie strony. Ponieważ ten sam kod może być używany zarówno w części klienckiej, jak i serwerowej programu, nie ma potrzeby powielania tej samej funkcjonalności. W rezultacie czas i koszty opracowywania są skrócone;
- Ponowne wykorzystanie kodu znacznie ułatwiło tworzenie aplikacji mobilnych. Kod, który został napisany podczas tworzenia strony, może zostać w przyszłości wykorzystany do stworzenia aplikacji mobilnej.

TESTOWANIE

Celem testowania jest dostarczenie informacji o poziomie spójności pomiędzy elementami testu a wymaganiami dostarczonymi przez Welly. Informacje te będą dostarczane poprzez środki kontroli jakości podjęte w celu kontroli jakości i defektów zarejestrowanych w wyniku testów i niezgodności. Wszystkie defekty będą miały atrybut Priorytetowy, który ustalany jest na podstawie analizy wagi i częstotliwości korzystania z funkcjonalności przez klienta oraz ryzyka jej wystąpienia w eksploatacji. Wszystkie defekty będą miały atrybut wagi, który jest określany przez analizę zakresu, w jakim defekt może wpływać na program.
Kontrola jakości zostanie przeprowadzona przez uczestnika odpowiedzialnego za Testy Ręczne na poziomie Systemu i Integracji.
Jeden z etapów testowania będzie również polegał na porównaniu zachowania istniejącego programu w środowisku produkcyjnym z jego nową wersją.

Ponieważ cały proces tworzenia oprogramowania jest podzielony na sprinty (2 tygodnie każdy), czynności testowe zostaną podzielone w ten sam sposób, aby dopasować się do procesu rozwoju.
Główny przebieg sprintu przedstawiono na poniższym schemacie:



Podczas testu wykonywane są następujące zadania:

1. Historie użytkowników \ Analiza ulepszeń;
2. Definicja testowanych funkcji;
3. Identyfikacja cech, które nie podlegają weryfikacji;
4. Określanie wymaganej głębokości testów w oparciu o złożoność funkcjonalności, które należy opracować;
5. Określenie głębokości testów regresyjnych wymaganych na koniec projektu, w oparciu o zakres projektu;
6. Ocena działań kontroli jakości projektu, która powinna obejmować następujące działania:

• Opracowywanie testów i implementacja przypadków testowych;

• Smoke, UI, testowanie funkcjonalne w oparciu o liczbę obsługiwanych konfiguracji;

• Testy potwierdzające oparte na złożoności opracowanej funkcjonalności i pomiarach projektowych;

• Sesje testowania badań;

• Testy regresyjne oparte na złożoności opracowywanych funkcjonalności i pomiarach projektu.

Przypadki testowe zostaną utworzone w Azure DevOps, które powinny zawierać następujące informacje:

1. ID
2. Title
3. Summary
4. Assigned To (ім'я)

5. State (Design/Ready/Closed)
6. Priority (1-Critical; 4-Low)
7. Area (AdvancedMD/Platform SWAT)
8. Iteration
9. Steps
10. Expected result
11. Tested User Story
12. Attachments (dodatkowe)
13. Comments (dodatkowe)

Do testów zostaną również wykorzystane następujące usługi i technologie:

Jmeter, Fiddler, Swagger, Postman