Plan projektu

Wersja 1.00.

<Strona Internetowa cyfrowydochód.pl>

Jakub Zając | Piotr Śliwa

[Rok]

Spis treści

[1. Wstęp 1](#_Toc163110304)

[2. Opis projektu 1](#_Toc163110305)

[3. Struktura organizacyjna projektu 1](#_Toc163110306)

[4. Role i odpowiedzialność 1](#_Toc163110307)

[5. Harmonogram projektu 1](#_Toc163110308)

[6. Kosztorys projektu 1](#_Toc163110309)

[7. Zasoby projektu 1](#_Toc163110310)

[8. Standardy i narzędzia w projekcie 1](#_Toc163110311)

[9. Procesy zarządzania 1](#_Toc163110312)

[9.1 Plan zarządzania konfiguracją 1](#_Toc163110313)

[9.2 Plan zarządzania ryzykiem 1](#_Toc163110314)

[9.3 Plan zarządzania testami 1](#_Toc163110315)

1. Wstęp   
cyfrowydochód.pl – strona internetowa na której użytkownicy będą mogli:  
a) zakupić jakościowe ebooki dotyczące zarabiania przez internet  
b) spędzać czas na przyjaznej interfejsowo stronie internetowej  
c) zarządzać nowymi ebookami, statystykami, przeglądać przychody strony internetowe (admini)

2. Opis projektu   
CyfrowyDochód.pl to aplikacja internetowa składająca się z backendowego WebAPI napisanego całościowo w języku programowania Java (Spring, SpringBoot, Hibernate), mającego połączenie z bazą danych MySQL, a także z frontendową aplikacją napisaną w TypeScript(React, TailwindCSS, ShadCN/UI). Zamysł aplikacji jest prosty: Chcemy stworzyć przyjazną dla nowych użytkowników aplikację internetową, która zachęci ludzi do nauki sposobów zarabiania przez internet, aby mogli odmienić swoje życie.

3. Struktura organizacyjna projektu

Zespół Backend: Jakub Zając, pomocnik Piotr Śliwa

Zespół FrontEnd: Piotr Śliwa.

#### Przepływ danych między zespołami

1. **Zespół Backend**:
   * Odpowiada za tworzenie i utrzymanie WebAPI.
   * Współpracuje z zespołem Frontend w celu integracji API z interfejsem użytkownika.
   * Dostarcza dokumentację API dla zespołu Frontend.
2. **Zespół Frontend**:
   * Odpowiada za tworzenie i utrzymanie interfejsu użytkownika.
   * Współpracuje z zespołem Backend w celu integracji interfejsu z API.
   * Zgłasza wymagania dotyczące API do zespołu Backend.

#### Zasady organizacji spotkań

* Cotygodniowe spotkania statusowe z udziałem obu zespołów.
* Codzienne stand-upy w ramach metodyki Agile/Scrum.
* Miesięczne retrospektywy i planowanie sprintów.

4. Role i odpowiedzialność

1. **Project Manager**:
   * Zarządzanie projektem, koordynacja zespołów.
   * Monitorowanie postępów, raportowanie do interesariuszy.
   * Rozwiązywanie problemów i zarządzanie ryzykiem.
2. **Lead Backend Developer** (Jakub Zając):
   * Projektowanie i implementacja backendowego WebAPI.
   * Nadzór nad pracą zespołu Backend.
   * Współpraca z zespołem Frontend.
3. **Backend Developer** (Piotr Śliwa):
   * Wspieranie implementacji WebAPI.
   * Pisanie kodu, testowanie, debugowanie.
4. **Lead Frontend Developer** (Piotr Śliwa):
   * Projektowanie i implementacja interfejsu użytkownika.
   * Nadzór nad pracą zespołu Frontend.
   * Współpraca z zespołem Backend.
5. **QA Specialist**:
   * Planowanie i przeprowadzanie testów.
   * Zarządzanie błędami, raportowanie wyników testów.

5. Harmonogram projektu

Harmonogram projektu został szczegółowo opisany w pliku proj\_cyfrowydochod.pl.mpp.  
Wykres Gantta:A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

6. Kosztorys projektu

|  |  |  |  |
| --- | --- | --- | --- |
| **Rola** | **Liczba godzin** | **Stawka godzinowa[PLN]** | **Koszt całkowity**  **[PLN]** |
| Project Manager | 100 | 100 | 10,000 |
| Lead Backend Dev | 200 | 120 | 24,000 |
| Backend Dev | 150 | 100 | 15,000 |
| Lead Frontend Dev | 200 | 120 | 24,000 |
| QA Specialist | 100 | 90 | 9,000 |
| **Łącznie** | **750** | **-** | **82,000** |

7. Zasoby projektu

 **Sprzęt**: Komputery, serwery, urządzenia mobilne do testowania responsywności.

 **Oprogramowanie**:

* IDE: IntelliJ IDEA (Java), Visual Studio Code (JavaScript/TypeScript).
* Baza danych: MySQL.
* Systemy kontroli wersji: Git, GitHub.
* Narzędzia do zarządzania projektem: JIRA, Trello.
* Narzędzia do komunikacji: Slack, Zoom.
* Licencje: IntelliJ IDEA Ultimate, MySQL Enterprise.

8. Standardy i narzędzia w projekcie

 **Standardy kodowania**:

* Java: Google Java Style Guide.
* JavaScript/TypeScript: Airbnb JavaScript Style Guide.

 **Narzędzia**:

* Backend: Spring Boot 2.6, Hibernate 5.6.
* Frontend: React 18, TypeScript 4.5, TailwindCSS 3.0, ShadCN/UI 1.0.
* Testowanie: JUnit 5 (backend), Jest (frontend).

9. Procesy zarządzania

9.1 Plan zarządzania konfiguracją

* **Narzędzie zarządzania konfiguracją**: Git, GitHub.
* **Zasady tagowania dokumentów**:
  + Nazewnictwo commitów: [type] short description (type: feat, fix, docs, style, refactor, test, chore).
  + Tagowanie wersji: vX.Y.Z (e.g., v1.0.0).

9.2 Plan zarządzania ryzykiem

 **Cele zarządzania ryzykiem**:

* Identyfikacja potencjalnych ryzyk na wczesnym etapie.
* Ocena i priorytetyzacja ryzyk.
* Planowanie i wdrażanie strategii mitygacyjnych.

 **Definicje pojęć**:

* **Ryzyko projektu**: Wydarzenie, które może negatywnie wpłynąć na realizację celów projektu.
* **Waga ryzyka**: Ocena potencjalnego wpływu ryzyka na projekt.
* **Stopień ryzyka**: Prawdopodobieństwo wystąpienia ryzyka pomnożone przez jego wagę.

 **Struktura bazy ryzyk**:

* Identyfikator ryzyka.
* Opis ryzyka.
* Prawdopodobieństwo wystąpienia.
* Waga ryzyka.
* Stopień ryzyka.
* Strategia mitygacyjna.
* Status ryzyka.

9.3 Plan zarządzania testami

 **Cele zarządzania testami**:

* Zapewnienie wysokiej jakości produktu końcowego.
* Wykrycie i naprawa błędów na wczesnym etapie.
* Weryfikacja zgodności produktu z wymaganiami.

 **Zasady projektowania testów**:

* Testowanie jednostkowe (Unit Testing).
* Testowanie integracyjne (Integration Testing).
* Testowanie end-to-end (E2E Testing).
* Testowanie wydajnościowe (Performance Testing).

 **Proces zarządzania błędami**:

* Wykrycie błędu.
* Zgłoszenie błędu w systemie zarządzania błędami (JIRA).
* Analiza błędu.
* Naprawa błędu.
* Retest.
* Zamknięcie zgłoszenia błędu.

 **Definicje i lista statusów błędu**:

* **New**: Nowy błąd, czeka na analizę.
* **In Progress**: Błąd jest naprawiany.
* **Fixed**: Błąd został naprawiony.
* **Retest**: Błąd czeka na ponowne testowanie.
* **Closed**: Błąd został rozwiązany i zamknięty.
* **Rejected**: Błąd został odrzucony jako nieważny.

 **Struktura bazy błędów**:

* Identyfikator błędu.
* Opis błędu.
* Status błędu.
* Priorytet błędu.
* Kto zgłosił.
* Kto naprawia.
* Data zgłoszenia.
* Data naprawy.

 **Cykle testowe**:

* Sprintowe testowanie jednostkowe i integracyjne.
* Testowanie regresji na koniec każdego sprintu.
* Testy końcowe przed wydaniem wersji produkcyjnej.

9.4 Plan zapewnienia jakości

 Role QA w procesie wytwarzania oprogramowania

* QA Manager: Odpowiedzialny za nadzór nad procesami QA, zarządzanie zespołem QA, planowanie i realizację zadań QA
* QA Engineer: Przeprowadza testy zgodnie z planem testów, raportuje błędy, współpracuje z developerami w celu rozwiązania wykrytych błędów.

 **Metodologie QA**:

* Testowanie ręczne
* Testy jednostokowe
* Testy systemowe
* Testy wydajnościowe
* Testy bezpieczeństwa

 **Elementy podlegające kontroli QA**:

* Kod źródłowy
* Wymagania
* Przypadki testowe.

 **Lista dokumentów Tworzonych przez QA**

* Plan testów
* Przypadki testowe
* Raporty testów
* Rejestr błędów

· **Zakres odpowiedzialności osób realizujących zadania QA**:

* QA Manager
  + Opracowanie i utrzymanie planu QA
  + Nadzór nad realizacją zadań QA
  + Raportowanie stanu jakości projektu
* QA Engineer
  + Tworzenie test case-ów
  + Wykonywanie testów ręcznych
  + Raportowanie i zarządzanie błędami