**Dokument wymagań projektowych**

**Nazwa projektu: Rat-a-tat**

**Autorzy: Aleksander Burkowski**

**Data: 17.05.2022**

**Historia zmian**

|  |  |  |  |
| --- | --- | --- | --- |
| Wersja | Data | Opis | Autorzy |
| 1.0 | 10.05.2022 | Rozpoczęcie pracy nad dokumentem wymagań projektowych | Aleksander Burkowski |
| 1.1 | 17.05.2022 | Określenie pełnej funkcjonalności projektu. Rozszerzenie „Granic projektu” | Aleksander Burkowski |

**Spis treści**

[**Historia zmian** 1](#_Toc103177408)

[**1. Elementy składowe projektu (produkty projektu)** 2](#_Toc103177409)

[**2. Granice projektu** 2](#_Toc103177410)

[**3. Lista wymagań funkcjonalnych** 2](#_Toc103177411)

[**4. Lista wymagań niefunkcjonalnych** 4](#_Toc103177412)

[**5. Kryteria akceptacji projektu dla I semestru prac** 4](#_Toc103177413)

[**6. Mierzalne wskaźniki wdrożeniowe** 5](#_Toc103177414)

[**7. Kryteria akceptacji projektu dla II semestru prac** 5](#_Toc103177415)

[**9. Ryzyka projektowe** 5](#_Toc103177416)

[**10. Kamienie milowe** 6](#_Toc103177417)

**1. Elementy składowe projektu (produkty projektu)**

Elementy programistyczne:

* aplikacja serwerowa w języku C# oparta o platforme ASP.NET Core
* aplikacja po stronie klienta z wykorzystaniem biblioteki React.js
* instancja serwera bazy danych oparta o silnik MSSQL
* integracja systemu płatności internetowej

Elementy nieprogramistyczne:

* specyfikacja funkcjonalna
* prototyk projektu wykonany przy użyciu aplikacji Figma
* dokumentacja użytkownika
* grafiki rewersów kart

**2. Granice projektu**

Serwis umożliwiać ma grę w rat-a-tat. Cechy aplikacji na których nam zależy to estetyka, wydajność i sprawność, zbiegać się one będą z podstawowymi i największymi oczekiwaniami potencjalnego klienta. Aplikacja oferować ma również dodatkowe funkcjonalności, w tym :

* Tworzenie i personalizacja konta klienta
* Lista znajomych
* Gra z komputerem
* Turnieje
* Ranking graczy
* Chat

Miktotransakcje w aplikacji odbywać się będą za pomocą zewnętrznego panelu płatności elektronicznych. Aplikacja hostowana będzie poprzez usługę AWS Elastic Beanstalk.

Z racji wykorzystywanych przez nas technologii, jesteśmy zmuszeni wymagać od użytkownika przeglądarki obsługującej jQuery w wersji 1.6.4, lub nowszej. Na ten moment przeglądarki oraz ich wersje, wspierające jquery to:

|  |  |
| --- | --- |
| Przeglądarka | Wersja |
| Chrome | 101.0.4951.67 lub 100+ |
| Mozilla Firefox | 100.0 lub 99.0+ |
| Opera | 12.15 lub 11.0+ |
| Microsoft Edge | 100.0.1185.39 lub 99.0+ |
| Microsoft Internet Explorer | 11.0 + |
| Safari | 15.3 lub 14.0+ |

Lista graczy, którą będziemy obsługiwać nie jest nam do tej pory znana, serwis AWS umożliwia jednak swobodne skalowanie aplikacji.

Wymagać będziemy od klientów tworzenia kont, a co za tym idzie podawania nam swoich danych.  
Każdy klient będzie posiadać swój ranking, liste znajomych. Na bazie doświadczenia gracza dobierany będzie dla niego stół.  
Wszystkie informacje o kliencie zapisywane będą w bazie danych.

Zdecydowaliśmy się na stworzenie aplikacji webowej ponieważ :

* Nie chcieliśmy zmuszać klientów do pobierania aplikacji, zależy nam na dostępności aplikacji
* Technologia z której korzystamy daje nam duże możliwości skalowania
* Rozwój aplikacji nie będzie zmuszać klientów do pobierania aktualizaji.

**3. Lista wymagań funkcjonalnych**

* Use case 1 – Rejestracja użytkownika  
  Użytkownik planuje założyć konto w aplikacji. Wciska przycisk “Utwórz nowe konto”, który przenosi go do formularza rejestracji. Wpisuje odpowiednie dane oraz klika przycisk “Utwórz konto”. Jeśli wszystkie dane sa poprawne, otrzymuje maila z linkiem aktywacyjnym. Po wejściu w link, konto staje się aktywne I może się na nie zalogować.
* Use case 2 – Logowanie użytkownika  
  Użytkownik chce się zalogować do serwisu. Wciska przycisk “Zaloguj się”, który przenosi go do panelu logowania. Wpisuje swoje dane I wciska przycisk “Zaloguj się”. Zostaje przeniesiony do strony głównej serwisu. W przypadku wpisania błędnych danych, zostanie o tym poinformowany.
* Use case 3 – Wylogowanie  
  Użytkownik chce się wylogować. Rozsuwa liste z górnego panelu i wybiera opcję “Wyloguj się”. Użytkownik zostaje wylogowany.
* Use case 4 – Resetowanie hasła  
  Użytkownik zapomniał hasła do swojego konta. Klika przycisk “Zaloguj się”, który przenosi go do panelu logowania. W panelu logowania wciska przycisk “Nie pamiętam hasła”, który przenosi go do formularza. W formularzu podaje email swojego konta. Na podany adres zostaje wysłany link do zmiany hasła. Użytkownik dwa razy wprowadza nowe hasło I klika akceptuj. Jego hasło zostaje zmienione i użytkownik zostaje odesłany do panelu logowania.
* Use case 5 – Personalizacja konta  
  Zalogowany użytkownik chce dokonać zmian na swoim koncie. Rozsuwa liste z górnego panelu i wybiera opcję “Ustawienia konta”, która przenosi go do panelu personalizaji konta.  
  Użytkownik ma do wyboru : zmianę rewersu, ikony profilu, zmiane nazwy użytkownika.  
  Po wybraniu zmian wciska przycisk “Zapisz”, po czym zmiany zostają zastosowane.
* Use case 6 – Lista znajomych

Zalogowany użytkownik chce wejść do swojej listy znajomych. Rozsuwa liste z górnego panelu I wybiera opcję “Znajomi”, która przenosi go do listy znajomych użytkownika.

* Use case 7 – Dodanie nowego znajomego  
  Zalogowany użytkownik chce dodać nowego znajomego. Rozsuwa liste z górnego panelu i wybiera opcję “Znajomi”, która przenosi go do listy znajomych użytkownika. Wciska przycisk “Dodaj”, otwarte zostaje okno. Użytkownik podaje nick znajomego oraz wciska przycisk “Dodaj znajomego”. Gracz o podanym nicku otrzymuje powiadomienie o zaproszeniu. Po zaakceptowaniu gracze trafiają do list znajomych.
* Use case 8 – Stworzenie nowego stołu  
  Zalogowany użytkownik chce założyć nowe stół do gry. Na stronie głównej wybiera przycisk “Stwórz nową grę”, otwarte zostaje okno z możliwością wyboru opcji rogrywki. Gracz ustawia nazwę stołu, czas na ruch i wybiera czy stół ma być dostępny tylko dla znajomych oraz czy rozgrywka ma być rankingowa. Po uzupełnieniu danych wciska przycisk “Stwórz nową grę” i zostaje przeniesiony do widoku stołu.
* Use case 9 – Dołączenie do istniejącego stołu  
  Zalogowany użytkownik chce dołączyć do istniejącego już stołu. Wybiera istniejący już stół ze strony głównej. Zostaje przeniesiony do wybranego pokoju, gdzie wciska przycisk “Dołącz”. Gracz został dodany do wybranego stołu.
* Use case 10 – Rozpoczęcie rozgrywki  
  Zalogowany użytkownik chce zagrać w rat-a-tat Cat. Wykonuje czynności z Use case 8 lub Use case 9. Gracz po dołączeniu do stołu, wciska przycisk “Gotowy”. Gracz czeka na pozostałych zawodników. Gdy wszyscy zawodnicy są gotowi, rozgrywka się zaczyna.
* Use case 11 – Zagranie karty  
  Gracz bierze udział w rozgrywce i chce zagrać karte. Wybiera kartę z listy. Karta zostaje rzucona. W przypadku wyrzucenia złej karty, gracz dostaje z powrotem rzuconą kartę oraz jedną dodatkową.
* Use case 12 – Wybór doboru karty  
  Gracz bierze udział w rozgrywce i chce dobrać karte. Ma do wyboru wzięcie ostatniej rzuconej karty lub wzięcie karty z góry tali. Po dokonanym wyborze gracz otrzymuje karte.
* Use case 13 – Zagranie karty specjalnej  
  Gracz bierze udział w rozgrywce i rzuca kartę specjalną. W zależności od wyrzuconej karty zostają mu przedstawione opcje ruchu. Wybiera jedną z opcji po czym wykonany jest efekt karty.
* Use case 14 – Lista rankingowa  
  Zalogowany użytkownik chce zobaczyć jak wygląda aktualny ranking graczy. Rozsuwa liste z górnego panelu i wybiera opcję “Ranking”, która przenosi go do widoku rankingu.
* Use case 15 – filtrowanie stołów  
  Zalogowany użytkownik chce pofiltrować stoły według danych kryteriów. Na głównej stronie wybiera interesujące go tryby rozgrywki i ustawienia stołu. Pokoje zostają przefiltowane tak aby spełniać podane kryteria.
* Use case 16 – Chat  
  Gracz bierze udział w rozgrywce i chce skorzystać z funkcjonalności komunikatora. W prawym dolnym roku znajduje się okno czatu. Użytkownik pisze swoją wiadomość po czym wciska przycisk wyślij. Pozostali gracze otrzymują wiadomość.

**4. Lista wymagań niefunkcjonalnych**

* Aplikacja ma być dostępna na wszystkich nowych, popularnych przeglądarkach
* Przewidywalna maksymalna ilość graczy to 500 osób
* Maksymalny czas odpowiedzi aplikacji na zapytanie użytkownika wynosi sekunde
* Użytkownicy będą mogli zgłaszać błędy dotyczące systemu poprzez poczte internetową
* Aplikacja będzie korzystać z protokołu HTTPS w celach zwiększenia bezpieczeństwa
* System oferować będzie wprowadzenie do gry, samouczek.
* Serwis ma być przejrzysty i prosty.

**5. Kryteria akceptacji projektu dla I semestru prac**

* Wymagane
  + Dokument wizji
  + Dokument wymagań projektowych
  + Prototyp interfejsu
  + Wdrożenie podstawowych funkcjonalności produktu: Założenie konta, logowanie, tworzenie / dołączenie do stołu
* Oczekiwane
  + Stworzenie samouczka dla graczy
  + Doprowadzenie strony do przewidzianego wyglądu
  + Działająca logika gry rat-a-tat Cat

**6. Mierzalne wskaźniki wdrożeniowe**

* System zostanie udostępniony poprzez usługę AWS elastic beanstalk
* Aplikacja zostanie poddana testom obciążeniowym, a otrzymane przez nie wyniki będą spełniać założenia projektu
* W ramach serwisu klienci będą mogli zagrać w rat-a-tat Cat
* Z końcem drugiego semestru pierwsi użytkownicy będą mogli zakładać konta i grać w rat-a-tat Cat korzystając z naszej aplikacji.

**7. Kryteria akceptacji projektu dla II semestru prac**

* Wymagane
  + Działająca logika gry rat-a-tat Cat
  + Dodanie reszty funkcjonalności przewidzianych w projekcie
  + Udostępnienie aplikacji poprzez usługę AWS
* Oczekiwane
  + Zwiększenie maksymalnej ilości graczy
  + Raporty z testów obciążeniowych

**8. Organizacja pracy zespołu**

Aleksander Burkowski - backend (C#), praca nad dokumentacją

Piotr Jakub Dębski - frontend (React.js, Typescript), jira manager i projektowanie interfejsu

Krzysztof Pajsert - frontend (React.js, Typescript) kontakt z klientem, projektowanie interfejsu

Paweł Radny - backend (C#), git manager

Komunikacja z klientem odbywa się poprzez aplikacje Microsoft Teams, w spotkaniu biorą udział wszyscy członkowie zespołu. Na każdym spotkaniu omawiane będą nowo dodane funkcje oraz ewentualne uwagi klienta.

W ramach projektu korzystać będziemy z :

* Repozytorium Git uczelni UAM
* Oprogramowania Jira wspomagającego zarządzanie projektem
* Środowisko programistyczne Microsoft Visual Studio Code
* program Figma służący do projektowania i prototypowania aplikacji webowej

Zespół przyjął metodyke pracy Scrum.  
W organizacji pracy pomoże nam narzędzie Jira, na którym między innymi będą tworzone Taski i Sprinty. Taski będa tworzone przed rozpoczęciem Sprintu ale także w trakcie w przypadku nie przewidzenia jakiegoś problemu wcześniej.  
Praca prowadzona będzie w trybie iteracyjnym, tygodniowe sprinty pozwolą na regularne dostarczanie poszczególnych części projektu aż do finalnego rozwiązania.

**9. Ryzyka projektowe**

* Aplikacja zostanie wdrożona przy pomocy serwisu AWS, oszacowanie zasobów wymaganych do płynnego działania aplikacji przy dużym obciążeniu, wybiega poza nasze możliwości.
* Implementacja systemu opartego o websocket’y może być dla nas problematyczna, ponieważ nie posiadamy wcześniejszego doświadczenia z tą technologią. To ryzyko uważamy jednak za mniej istotne

**10. Kamienie milowe**

* Stworzenie mockupu projektu w Figmie
* Dodanie funkcjonalności rejestracji oraz logowania
* Tworzenie / dołączanie do stołów
* Stworzenie modelu gry rat-a-tat Cat
* Dodanie modułu profila użytkownika
* Dodanie listy znajomych
* Implementacja modułu chatu
* Hosting na usłudze AWS