|  |
| --- |
|  |
| Dokumentacja testów ePrzychodnia |
|  |
|  |
| **Jan Borowski, Maciej Graszek, Łukasz Wieczorek** |
| **2013-06-06** |

|  |
| --- |
| Wersja 2.1 |

**I. Wstęp**

1.Wprowadzenie

Celem przedsięwzięcia jest zbudowanie systemu do internetowej rejestracji do przychodni medycznej służącego do zarządzania wizytami i zabiegami oferowanymi przez przychodnię. Telefoniczna, bądź osobista rejestracja pacjentów jest czasochłonna i uciążliwa dla pacjentów. Tworzony system ma umożliwiać rejestrację pacjentów na wizyty i zabiegi w czasie zdefiniowanym przez lekarzy pracujących w poradniach przychodni.

2. Cele

Testy mają za zadanie przedstawić wydajność i funkcjonalność wybranych metod wykonywanych przez użytkownika, które należą do zbioru najczęściej wykonywanych akcji.

Z pomocą uzyskanych wyników, można wykonać zadania umożliwiające poprawę wydajności aplikacji.

3. Podejście

Strategia testów polega na wykonywaniu kilka tysięcy razy tej samej operacji pod rząd. Ma to na celu przetestować możliwości aplikacji i jej części serwerowej. Tym sposobem będzie można zrealizować kolejne kroki w celu poprawy efektywności aplikacji.

**II. Opis testów**

1. Testowany obiekt

Obiektem na którym wykonywane są testy jest aplikacją webową o nazwie "ePrzychodnia".

1. Aplikacja

ePrzychodnia jest aplikacją służącą do pośredniej komunikacji pomiędzy lekarzem, a konkretnych pacjentem. Aplikacja umożliwia pacjentowi zapisanie się do konkretnego lekarza w terminie odpowiadającym obu stronom. System informuje lekarza o potencjalnej wizycie, oraz pilnuje, by lekarz był na bieżąco z informacjami z nią związaną.

Po wizycie lekarz może zaproponować kolejne spotkanie w postaci wizyty lub zabiegu, w którym uczestniczy przynajmniej kilku lekarzy.

2. Funkcjonalność do przetestowania  
Aby można było w pełni wykonać testy zamawianej wizyty, należało wypełnić je poniższymi testami:

* tworzenie konta użytkownika
* logowanie się użytkownika
* przejście do panelu i zamówienie wizyty
* usuwanie wizyt
* sprawdzanie podstawowej walidacji danych

**III. Zarządzanie projektem testowym**

1. Zespół projektowy

* Maciej Graszek - dokumentacja testów
* Jan Borowski - testy funkcjonalne i jednostkowe
* Łukasz Wieczorek – testy obciążenia

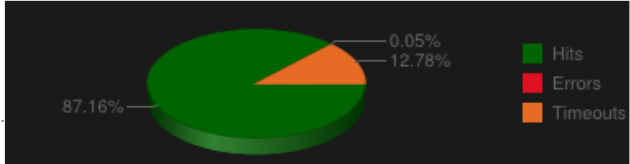
**IV. Testy**

1. Testy obciążenia

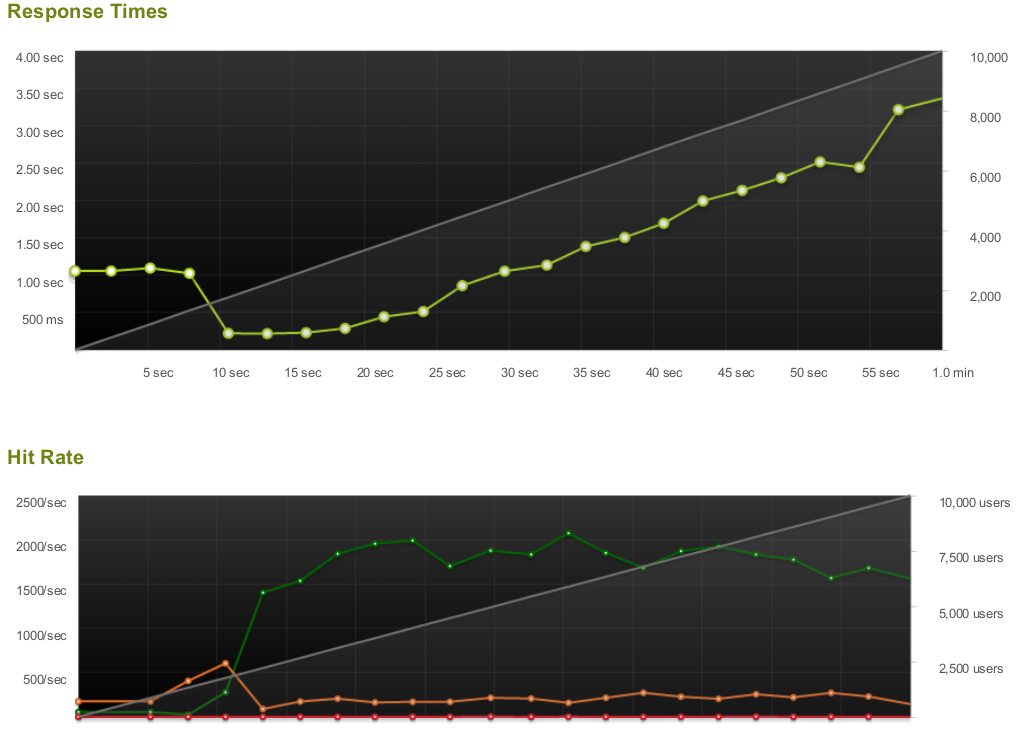
1.1. Testy zostały przeprowadzone z użyciem funkcjonalności oferowanej przez aplikację <https://www.blitz.io/>. Testowana aplikacja została uruchomiona w serwisie <https://www.heroku.com/>.

|  |  |
| --- | --- |
| ID: | **TO\_HOME** |
| Nazwa: | **Wyświetlenie strony startowej** |
| **Opis:** | |
| Wyświetlenie strony startowej | |
| **Warunki początkowe:** | |
| Aplikacja dostępna w serwisie Heroku. Dostępność środków płatniczych w serwisie blitz.io. | |
| **Warunki końcowe:** | |
|  | |
| **Scenariusz Testu:** | |
| 1. Użytkownik loguje się na konto w serwisie blitz.io  2. Użytkownik wpisuje w polu kwerendy: „--pattern 1-10000:60 --region ireland <http://shielded-crag-2772.herokuapp.com/>”, gdzie „shielded-crag-2772„ to nazwa aplikacji w serwisie Heroku  3. Użytkownik odczytuje raport testu | |
| **Oczekiwany wynik:** | |
| Aplikacja działa poprawnie i responsywnie przy rosnącym obciążeniu (1 – 10000 użytkowników) | |
| **Wynik:** | |
| Wygenerowano 88775 żądań (88.83MB danych) w czasie 60 sekund.  12,78% żądań zakończyło się timeoutem. 0,05% żądań zakończyło się błędem. | |
| **Wnioski:** | |
| 12,84% zapytań zakończyło się niepowodzeniem. | |

Wyniki testu:



Rysunek Trafienia testu TO\_HOME

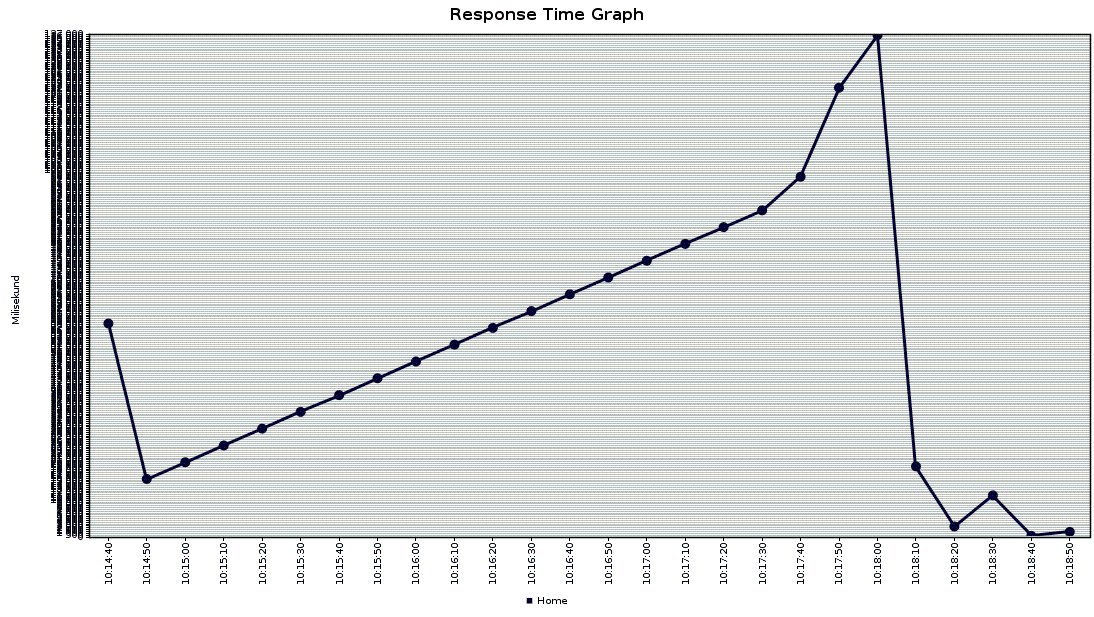


Rysunek Czas odpowiedzi i wskaźnik trafień testu TO\_HOME

1.2 Test 10000 wątków uruchamianych przyrostowo w 60 sekundach. Do poniższych testów wykorzystany został serwer WEBrick.

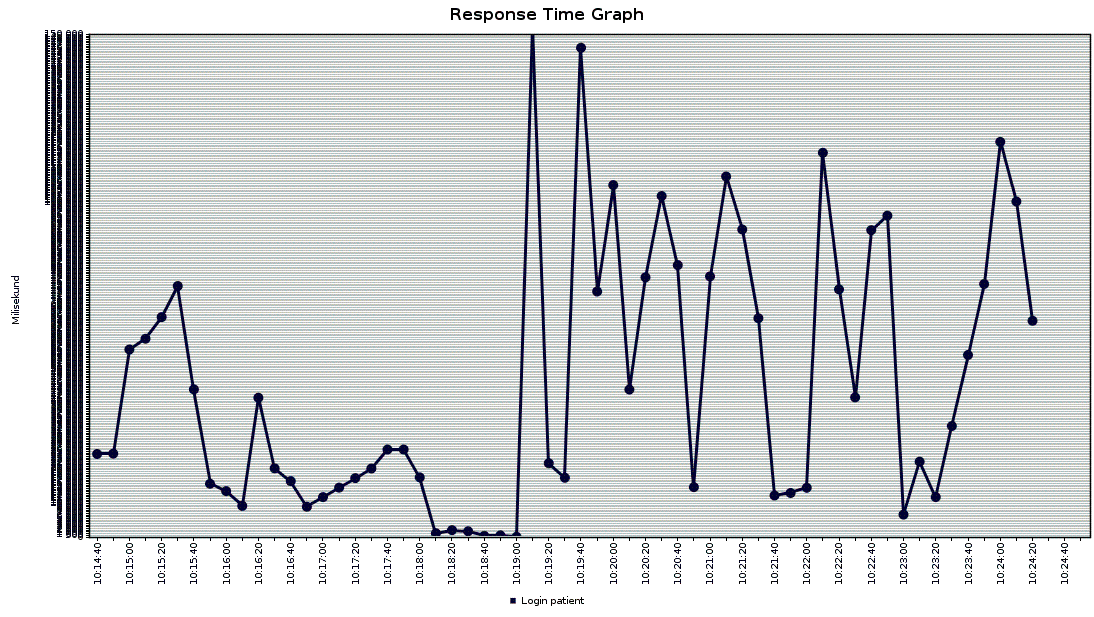
Testy wykonywane były na komputerze o następujących parametrach: Intel(R) Core(TM) i5 CPU M 580 @ 2.67GHz, 8GB RAM.

|  |  |
| --- | --- |
| ID: | **HOME** |
| Nazwa: | **Wyświetlenie strony startowej** |
| **Opis:** | |
| Wykonanie operacji GET dla pobrania zawartości strony startowej | |
| **Warunki początkowe:** | |
| Dostępność aplikacji pod adresem www. | |
| **Warunki końcowe:** | |
| Brak | |
| **Graficzny czas odpowiedzi:** | |
| Rysunek 3. | |
| **Maksymalny czas odpowiedzi:** | |
| 127 sekund | |
| **Oczekiwany wynik:** | |
| Aplikacja załaduje w pełni zawartość strony głównej przy rosnącym obciążeniu (1 – 10000 wyświetleń) | |
| **Procent błędnych wykonań testu** | |
| 98% | |
| **Wnioski:** | |
| Czas odpowiedzi przekracza dozwolony czas oczekiwania na ładowanie się zawartości strony. W pewnym momencie można zauważyć, jakoby serwer poddał się w krytycznym momencie i zignorował odpowiedź na ponad połowę wątków. Powód takiego zachowania jest nieznany. | |



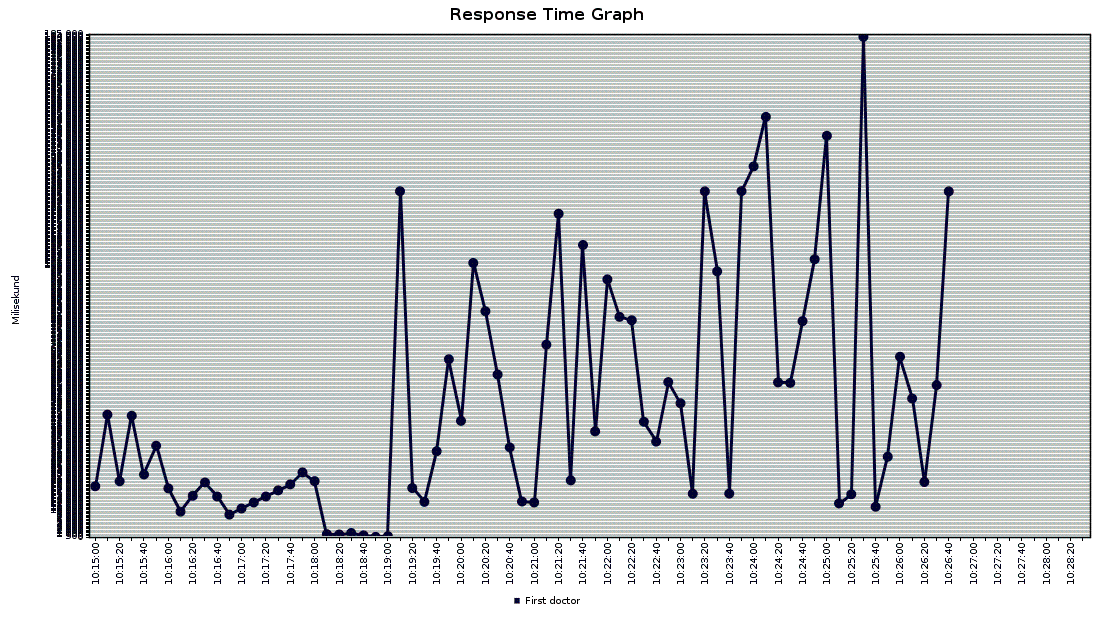
Rysunek Graf czasu odpowiedzi dla HOME

|  |  |
| --- | --- |
| ID: | **LOGIN PATIENT** |
| Nazwa: | **Logowanie się pacjenta** |
| **Opis:** | |
| Wykonanie operacji POST dla accounts/users/sign\_in user | |
| **Warunki początkowe:** | |
| Dostępność aplikacji pod adresem www. Istniejące konto na serwerze. | |
| **Warunki końcowe:** | |
| Brak | |
| **Graficzny czas odpowiedzi:** | |
| Rysunek 4 | |
| **Maksymalny czas odpowiedzi:** | |
| 109 sekund | |
| **Oczekiwany wynik:** | |
| Aplikacja przejdzie do panelu użytkownika zalogowanego przy rosnącym obciążeniu (1 – 10000 użytkowników) | |
| **Procent błędnych wykonań testu** | |
| 96% | |
| **Wnioski:** | |
| Czas odpowiedzi przekracza dozwolony czas oczekiwania na zalogowanie się pacjenta. Na poniższym wykresie można zauważyć nagłe wzrosty i skoki czasu odpowiedzi. Na podstawie tych danych można wywnioskować, że serwis nie jest przygotowany do próby logowania ponad 3-4 tys. użytkowników. | |



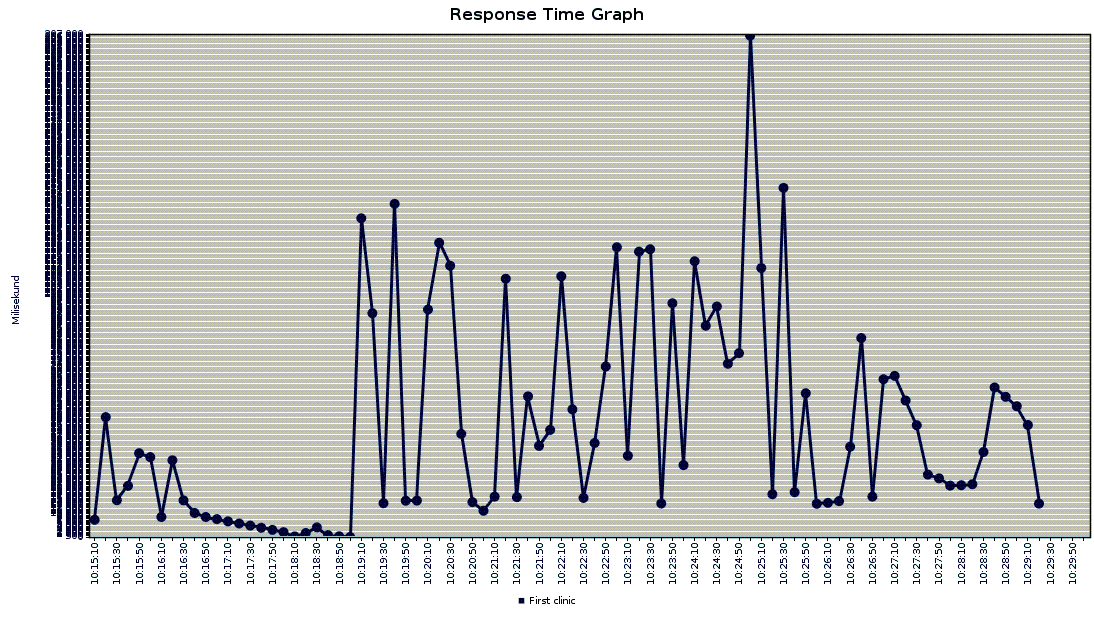
Rysunek Graf czasu odpowiedzi dla logowania

|  |  |
| --- | --- |
| ID: | **FIRST DOCTOR** |
| Nazwa: | **Szukanie najbliższej możliwej wizyty u lekarza** |
| **Opis:** | |
| Wykonanie operacji POST dla appointments/first\_doctor\_create date doctor | |
| **Warunki początkowe:** | |
| Dostępność aplikacji pod adresem www. Istniejące konto na serwerze, ze statusem pacjenta. Użytkownik musi być zalogowany. | |
| **Warunki końcowe:** | |
| Do systemu zostanie wprowadzony id lekarza. | |
| **Graficzny czas odpowiedzi:** | |
| Rysunek 5 | |
| **Maksymalny czas odpowiedzi:** | |
| 109 sekund | |
| **Oczekiwany wynik:** | |
| Aplikacja wyszuka najbliższy termin przyjęć u konkretnego lekarza przy rosnącym obciążeniu (1 – 10000 użytkowników) | |
| **Procent błędnych wykonań testu** | |
| 95% | |
| **Wnioski:** | |
| Czas odpowiedzi przekracza dozwolony czas oczekiwania na pobranie dostępnej wizyty. Na poniższym wykresie można zauważyć nagłe wzrosty i skoki czasu odpowiedzi. Powód niestabilności serwera przy takim dużym obciążeniu może być taki sam jak w przypadku logowania się użytkowników. | |



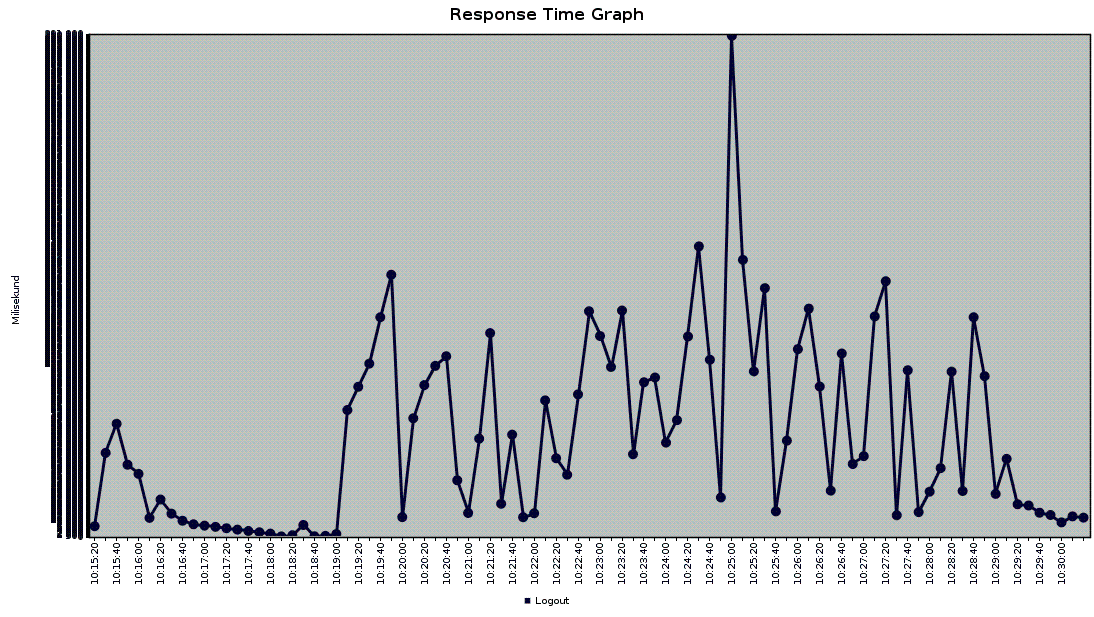
Rysunek Graf czasu odpowiedzi dla zapisania czasu prac

|  |  |
| --- | --- |
| ID: | **FIRST CLINIC** |
| Nazwa: | **Szukanie najbliższej możliwej wizyty w klinice** |
| **Opis:** | |
| Wykonanie operacji POST dla appointments/first\_clinic\_create date clinic | |
| **Warunki początkowe:** | |
| Dostępność aplikacji pod adresem www. Istniejące konto na serwerze, ze statusem pacjenta. Użytkownik musi być zalogowany. | |
| **Warunki końcowe:** | |
| Do systemu zostanie wprowadzony id kliniki. | |
| **Graficzny czas odpowiedzi:** | |
| Rysunek 6 | |
| **Maksymalny czas odpowiedzi:** | |
| 117 sekund | |
| **Oczekiwany wynik:** | |
| Aplikacja wyszuka najbliższy termin wizyty w konkretnej klinice przy rosnącym obciążeniu (1 – 10000 użytkowników) | |
| **Procent błędnych wykonań testu** | |
| 92% | |
| **Wnioski:** | |
| Czas odpowiedzi przekracza dozwolony czas oczekiwania na pobranie dostępnej wizyty. Na poniższym wykresie można zauważyć nagłe wzrosty i skoki czasu odpowiedzi, jednak nie są one tak duże jak w przypadku logowania się, pomimo wysyłania większej ilości danych. | |



Rysunek Graf czasu odpowiedzi na najbliższą wizytę w klinice

|  |  |
| --- | --- |
| ID: | **LOG OUT** |
| Nazwa: | **Wylogowanie się użytkownika** |
| **Opis:** | |
| Wykonanie operacji GET users/sign\_out | |
| **Warunki początkowe:** | |
| Dostępność aplikacji pod adresem www. Istniejące konto na serwerze. Użytkownik musi być zalogowany. | |
| **Warunki końcowe:** | |
| Brak. | |
| **Graficzny czas odpowiedzi:** | |
| Rysunek 7 | |
| **Maksymalny czas odpowiedzi:** | |
| 127 sekund | |
| **Oczekiwany wynik:** | |
| Powrót do strony głównej jako użytkownik niezalogowany. | |
| **Procent błędnych wykonań testu** | |
| 86% | |
| **Wnioski:** | |
| Czas odpowiedzi przekracza dozwolony czas oczekiwania na wylogowanie się pacjenta. Efekt jest zbliżony do uzyskanego podczas testu na oczekiwanie załadowania się strony głównej. Pomimo niewielkiej różnicy, ilość błędnych wykonań testu jest znacznie niższy. Opóźnienia związane z ładowaniem strony są niestety tak samo wysokie jak w przypadku innych zapytań. | |



Rysunek Graf czasu odpowiedzi na wylogowanie się użytkownika

|  |  |
| --- | --- |
| ID: | **FIND DOCTOR APPOINTMENTS** |
| Nazwa: | **Pobranie możliwych terminów wizyt** |
| **Opis:** | |
| Wykonanie operacji GET appointments | |
| **Warunki początkowe:** | |
| Dostępność aplikacji pod adresem www. Istniejące konto na serwerze. Użytkownik musi być zalogowany. | |
| **Warunki końcowe:** | |
| Pobrana zostaje lista możliwych wizyt u konkretnego lekarza. | |
| **Graficzny czas odpowiedzi:** | |
| Brak | |
| **Maksymalny czas odpowiedzi:** | |
| 116 sekund | |
| **Oczekiwany wynik:** | |
| Powrót do strony głównej jako użytkownik niezalogowany. | |
| **Procent błędnych wykonań testu** | |
| 73% | |
| **Wnioski:** | |
| Czas odpowiedzi jest wysoki i przekracza dozwoloną wartość. Należy zwrócić uwagę, iż uzyskiwana jest tutaj zawartość relacji bazodanowych i nie najprostszej konstrukcji. Wnioskować można, iż czas opóźnienie jest związane z błędem po stronie serwera, a nie samej aplikacji. | |

2. Testy funkcjonalne

Poniższe testy zostały wykonane przy pomocy Selenium IDE 1.10.0. Wszystkie testy zakończyły się powodzeniem.

|  |  |
| --- | --- |
| ID: | **TF\_LG** |
| Nazwa: | **Logowanie do systemu** |
| **Opis:** | |
| Logowanie pacjenta do systemu ePrzychodni | |
| **Warunki początkowe:** | |
| W bazie istnieje zarejestrowany użytkownik o parametrach takich jakie użyte są w testach. | |
| **Warunki końcowe:** | |
| Użytkownik zostanie zalogowany do sytemu | |
| **Scenariusz Testu:** | |
| 1. Użytkownik przechodzi na stronę startową aplikacji „ePrzychodnia”  2. Użytkownik przechodzi do strony logowania  3. Użytkownik podaje dane konta testowego (login, hasło)  4. Użytkownik przechodzi na stronę dostępnych wizyt | |
| **Wynik:** | |
| Pacjent loguje się do systemu. | |
| **Scenariusz alternatywny** | |
| 3A. Użytkownik podaje błędne dane  3A.1 Użytkownik podaje dane dla nieistniejącego konta  3A.2 System pokazuje informacje o nieprawidłowych danych  3A.3 Koniec przypadku testowego | |
| **Wnioski:** | |
| Test zakończył się bez błędów. System spełnia funkcjonalność | |

|  |  |
| --- | --- |
| ID: | **TF\_ZW** |
| Nazwa: | **Zamawianie wizyty** |
| **Opis:** | |
| Zamawianie wizyty przez pacjenta | |
| **Warunki początkowe:** | |
| W bazie istnieje zarejestrowany użytkownik o parametrach takich jakie użyte są w testach. | |
| **Warunki końcowe:** | |
| Użytkownik tworzy wizytę | |
| **Scenariusz Testu:** | |
| 1. Użytkownik loguje się do systemu  2. Użytkownik przechodzi na stronę dostępnych wizyt  3. Użytkownik wybiera dostępną pierwszą wizytę  4. Użytkownik przechodzi do strony swoich wizyt  5. Użytkownik potwierdza wcześniej umówioną wizytę | |
| **Wynik:** | |
| Pacjent umówiony na potwierdzoną wizytę | |
| **Wnioski:** | |
| Test zakończył się bez błędów. System spełnia funkcjonalność | |

|  |  |
| --- | --- |
| ID: | **TF\_UNPW** |
| Nazwa: | **Usuwanie niepotwierdzonej wizyty** |
| **Opis:** | |
| Usunięcie wizyty przez pacjenta | |
| **Warunki początkowe:** | |
| W bazie istnieje zarejestrowany użytkownik o parametrach takich jakie użyte są w testach. | |
| **Warunki końcowe:** | |
| Wizyta jest usunięta z bazy danych. | |
| **Scenariusz Testu:** | |
| 1. Użytkownik loguje się do systemu  2. Użytkownik przechodzi do strony swoich wizyt  3. Użytkownik usuwa wybraną wizytę.  4. Użytkownik zatwierdza usunięte wizyty.  5. System usuwa wizytę z bazy danych i informuje lekarza o zmianach. | |
| **Wynik:** | |
| Pacjent odwołuje wizytę. | |
| **Wnioski:** | |
| Test zakończył się bez błędów. System spełnia funkcjonalność | |

|  |  |
| --- | --- |
| ID: | **TF\_UPW** |
| Nazwa: | **Usuwanie potwierdzonej wizyty** |
| **Opis:** | |
| Usunięcie potwierdzonej wizyty przez pacjenta | |
| **Warunki początkowe:** | |
| W bazie istnieje zarejestrowany użytkownik o parametrach takich jakie użyte są w testach. | |
| **Warunki końcowe:** | |
| Wizyta jest usunięta z bazy danych. | |
| **Scenariusz Testu:** | |
| 1. Użytkownik loguje się do systemu  2. Użytkownik przechodzi do strony swoich wizyt  3. Użytkownik usuwa wybraną wizytę.  4. Użytkownik zatwierdza usunięte wizyty.  5. System informuje o braku możliwości usunięcia potwierdzonej wizyty. | |
| **Wynik:** | |
| Pacjent nie może odwołać potwierdzonej wizyty. | |
| **Wnioski:** | |
| Test zakończył się bez błędów. System spełnia funkcjonalność | |

|  |  |
| --- | --- |
| ID: | **TF\_UNW** |
| Nazwa: | **Umówienie na wizytę** |
| **Opis:** | |
| Pacjent umawia się na wizytę | |
| **Warunki początkowe:** | |
| W bazie istnieje zarejestrowany pacjent oraz lekarz z dostępną wizytą o parametrach takich jakie użyte są w testach. | |
| **Warunki końcowe:** | |
| Utworzony zostaje termin wizyty | |
| **Scenariusz Testu:** | |
| 1. Użytkownik loguje się do systemu  2. Użytkownik przechodzi do strony wyboru lekarzy.  3. Użytkownik wybiera konkretnego lekarza.  4. Użytkownik wybiera jeden spośród kilku możliwych wizyt (dat)  3. Użytkownik zatwierdza swoje przybycie. | |
| **Wynik:** | |
| Pacjent umówiony na potwierdzoną wizytę | |
| **Wnioski:** | |
| Test zakończył się bez błędów. System spełnia funkcjonalność | |

|  |  |
| --- | --- |
| ID: | **TF\_LITPP** |
| Nazwa: | **Logowanie i tworzenia planu pracy** |
| **Opis:** | |
| Lekarz loguje się do systemu i tworzy nowy plan pracy | |
| **Warunki początkowe:** | |
| W bazie istnieje zarejestrowany lekarz o parametrach takich jakie użyte są w testach. | |
| **Warunki końcowe:** | |
| Utworzony zostaje termin wizyty | |
| **Scenariusz Testu:** | |
| 1. Lekarz loguje się do systemu podając swój login i hasło.  2. Lekarz przechodzi do zakładki planu pracy.  3. Lekarz tworzy nowy plan pracy.  4. Lekarz podaje godziny oraz dni urzędowania  5. Lekarz zatwierdza nowy plan pracy.  6. System przyjmuje nowy plan pracy i interpretuje je jako nowe terminy możliwych wizyt.. | |
| **Wynik:** | |
| Lekarz stworzył nowy plan pracy, na który pacjenci mogą zapisywać swoje wizyty. | |
| **Wnioski:** | |
| Test zakończył się bez błędów. System spełnia funkcjonalność | |

|  |  |
| --- | --- |
| ID: | **TF\_BL** |
| Nazwa: | **Błędne logowanie** |
| **Opis:** | |
| Użytkownik podaje niepoprawne dane do logowania się | |
| **Warunki początkowe:** | |
| Brak. | |
| **Warunki końcowe:** | |
| Brak | |
| **Scenariusz Testu:** | |
| 1. Użytkownik podaje login i hasło  2. Użytkownik zatwierdza dane logowania.  3. System odrzuca możliwość zalogowania się.  3. System informuje użytkownika i adresie email lub haśle. | |
| **Wynik:** | |
| Użytkownik jest informowany o błędnych danych | |
| **Wnioski:** | |
| Test zakończył się bez błędów. System spełnia funkcjonalność | |

|  |  |
| --- | --- |
| ID: | **TF\_UPP** |
| Nazwa: | **Usunięcie planu pracy** |
| **Opis:** | |
| Lekarz usuwa swój plan pracy | |
| **Warunki początkowe:** | |
| Istnieje użytkownik o statusie lekarza, który posiada plan pracy. | |
| **Warunki końcowe:** | |
| Plan pracy lekarza zostaje usunięty z bazy danych | |
| **Scenariusz Testu:** | |
| 1. Lekarz przechodzi do zakładki planu pracy.  2. Lekarz usuwa wybrany plan pracy.  3. System pyta o potwierdzenie zmian.  4. Lekarz zatwierdza zmiany.  5. System informuje pacjentów o usuniętych wizytach. | |
| **Wynik:** | |
| Plan pracy zostaje usunięty z bazy danych. Utworzone wizyty łączące z planem pracy zostają usunięte. Pacjenci są informowani o zmianach | |
| **Wnioski:** | |
| Test zakończył się bez błędów. System spełnia funkcjonalność | |

3. Testy jednostkowe

Narzędzia wykorzystane podczas testów jednostkowych to:

* Rspec (2.13.1)
* Factory Girl(4.2.1),
* Shoulda Matchers(2.0.0)

Serwer aplikacji: WEBrick 1.3.1

|  |  |
| --- | --- |
| ID: | **TJ\_REMOVEAPPAPPOINTMENT** |
| Nazwa: | **Zmiana statusu potwierdzonej wizyty** |
| **Opis:** | |
| Zmiana statusu wizyty z potwierdzonej na niepotwierdzoną. Metoda uncorfmin! dla potwierdzonej wizyty | |
| **Warunki początkowe:** | |
| Istnieje potwierdzona wizyta przez lekarza i pacjenta. | |
| **Warunki końcowe:** | |
| Brak | |
| **Uzyskany wynik:** | |
| Wizyta jest nadal potwierdzona | |
| **Wynik:** | |
| System nie zmienił statusu wizyty. Potwierdzona wizyta może zostać odwołana, ale nie można zmienić jej statusu z potwierdzonej na niepotwierdzoną.. W przypadku odwołania wizyty, zaleca się usunięcie istniejącej w celu dania możliwości zamówienia tej wizyty innemu pacjentowi | |
| **Wnioski:** | |
| Test zakończył się bez błędów. System spełnia funkcjonalność | |

|  |  |
| --- | --- |
| ID: | **TJ\_REMOVEAPPOINTMENT** |
| Nazwa: | **Zmiana statusu potwierdzonej wizyty** |
| **Opis:** | |
| Zmiana statusu wizyty z niepotwierdzonej na potwierdzoną. Metoda corfmin! dla potwierdzonej wizyty | |
| **Warunki początkowe:** | |
| Istnieje niepotwierdzona wizyta przez lekarza i pacjenta. | |
| **Warunki końcowe:** | |
| Brak | |
| **Uzyskany wynik:** | |
| Wizyta zostaje potwierdzona | |
| **Wynik:** | |
| System zmienił statusu wizyty. Wizyta zostaje potwierdzona, a informacja o tym powinna zostać przekazana do obu stron. | |
| **Wnioski:** | |
| Test zakończył się bez błędów. System spełnia funkcjonalność | |

|  |  |
| --- | --- |
| ID: | **TJ\_APPOINTMENT** |
| Nazwa: | **Test wizyty** |
| **Opis:** | |
| Test modelu Appointment | |
| **Warunki początkowe:** | |
| Model posiada pola patient, confirmed\_at, date, start, end, created\_at | |
| **Warunki końcowe:** | |
| Brak | |
| **Scenariusz Testu:** | |
| 1. Testowane powiązanie modelu z modelami Patient, Clinic\_doctor  2. Testowana metoda can\_confirm? | |
| **Oczekiwany wynik:** | |
| Bez błędów | |
| **Wynik:** | |
| Bez błędów | |
| **Wnioski:** | |
| Test zakończył się bez błędów. | |
| ID: | **TJ\_USER** |
| Nazwa: | **Test modelu** |
| **Opis:** | |
| Test modelu User | |
| **Warunki początkowe:** | |
| Model posiada pola email, lastname, firstname, confirmed\_at | |
| **Warunki końcowe:** | |
| Brak | |
| **Scenariusz Testu:** | |
| 1. Testowana unikalność pola email  2. Testowana obecność pól password, first\_name, last\_name  3. Testowana metoda unconfirm  4. Testowana metoda full\_name  5. Testowana metoda active\_for\_authentication | |
| **Oczekiwany wynik:** | |
| Bez błędów | |
| **Wynik:** | |
| Bez błędów | |
| **Wnioski:** | |
| Test zakończył się bez błędów. | |

5. Dokumentacja zmian

5.1 Zmiana serwera, włączenie HTTPS

Testy polegały na wygenerowaniu 150 żądań danej akcji w czasie 6 sekund. Podane czasy w tabelkach są w milisekundach.

Wszystkie wykonano na sprzęcie:

Procesor: Intel Pentrium DualCore T4200 2GHz

Pamięć: 3 GB

Dysk: 320 GB

5.2 Zmiana serwera, włączenie HTTPS

Zmiana serwera aplikacji z WEBrick na Apache (Passenger) pozytywnie wpłynęła na czasy odpowiedzi. Na podstawie przeprowadzonych testów nie udało się stwierdzić znaczących różnic pomiędzy działaniem aplikacji z i bez HTTPS.

|  |  |  |  |
| --- | --- | --- | --- |
| WEBrick | | | |
| Strona | Średnia | Maksymalny czas | Procent błędów |
| Home | 27143 | 134300 | 29,33% |
| Login patient | 25753 | 63164 | 29,33% |
| First doctor | 15900 | 63156 | 26,00% |
| First clinic | 12705 | 63144 | 18,00% |
| Available appointments | 11720 | 55206 | 3,33% |
| All Appointments | 8229 | 22418 | 0,00% |
| Logout | 10104 | 17007 | 0,00% |
| Login doctor | 20620 | 34679 | 0,00% |
| Find doctor appointment | 10669 | 27986 | 0,00% |
| All work plans | 7312 | 9466 | 0,00% |
| RAZEM | 15015 | 134300 | 10,60% |

|  |  |  |  |
| --- | --- | --- | --- |
| Passenger | | | |
| Strona | Średnia | Maksymalny czas | Procent błędów |
| Home | 5366 | 14771 | 0,00% |
| Login patient | 15969 | 20571 | 0,00% |
| First doctor | 10891 | 19660 | 0,00% |
| First clinic | 7254 | 9448 | 0,00% |
| Available appointments | 3481 | 4038 | 0,00% |
| All Appointments | 6645 | 7591 | 0,00% |
| Logout | 9174 | 16747 | 0,00% |
| Login doctor | 16026 | 20665 | 0,00% |
| Find doctor appointment | 10269 | 19658 | 0,00% |
| All work plans | 5700 | 8253 | 0,00% |
| RAZEM | 9078 | 20665 | 0,00% |

|  |  |  |  |
| --- | --- | --- | --- |
| Passenger + HTTPS | | | |
| Strona | Średnia | Maksymalny czas | Procent błędów |
| Home | 7943 | 11353 | 0,00 |
| Login patient | 19000 | 23394 | 0,00 |
| First doctor | 9966 | 22987 | 0,00 |
| First clinic | 7283 | 17110 | 0,00 |
| Available appointments | 6232 | 7900 | 0,00 |
| All Appointments | 6533 | 7136 | 0,00 |
| Logout | 6973 | 10512 | 0,00 |
| Login doctor | 18564 | 22134 | 0,00 |
| Find doctor appointment | 9107 | 21296 | 0,00 |
| All work plans | 5979 | 15082 | 0,00 |
| RAZEM | 9758 | 23394 | 0,00 |

5.3 Cache

Użyto memcache oraz gemu Dalli(2.0.5) do integracji z tejże biblioteki z Ruby on Rails. Ku naszemu zaskoczeniu zastosowanie cache wpłynęło negatywnie na wyniki aplikacji.

|  |  |  |  |
| --- | --- | --- | --- |
| Memcached | | | |
| Strona | Średnia | Maksymalny czas | Procent błędów |
| Home | 27113 | 146034 | 29,33% |
| Login patient | 31862 | 63228 | 29,33% |
| First doctor | 16679 | 63150 | 22,67% |
| First clinic | 13522 | 57021 | 19,33% |
| Available appointments | 13607 | 63101 | 4,00% |
| All Appointments | 8470 | 34350 | 0,00% |
| Logout | 9653 | 27729 | 0,00% |
| Login doctor | 22993 | 36496 | 0,00% |
| Find doctor appointment | 10202 | 33223 | 0,00% |
| All work plans | 7591 | 10489 | 0,00% |
| RAZEM | 16169 | 146034 | 10,47% |

5.4 Indeksy

ActiveRecord automatycznie zakłada w warstwie abstrakcji indeksy na kluczach głównych i obcych. Następnie są one transformowane na realne indeksy w bazie danych. Zdecydowano, by nie usuwać żadnego z indeksów, jako, że każdy z nich odnosi się do pola, często używanego do wyszukiwania i modyfikowania danych. Indeksy te istnieją zarówno w bazie danych SQLite jak i PostgreSQL.

5.5 Frontend

Po stronie frontendu został wykorzystany framework Bootstrap. W aplikacji zostało wykorzystane kilka odpowiednich kontrolek oraz zestaw css. Dopisanych zostało kilka klas css, głównie wykorzystywanych w tabelach oraz przyciskach.

Ze względu na wystarczająco dobrą jakość wizualną i estetyczną, zdecydowano się nie ingerować inwazyjnie w bibliotekę Bootstrap jak i jego plik css, naniesione zostały tylko niewielkie zmiany.

Zrezygnowano również ze stosowania kodu javascript dla zachowania krótszego czasu ładowania stron.

6. Wnioski

Aplikacja od strony backendu działa poprawnie i nie zostały dostrzeżone w niej żadne problemy na poziomie wysokim lub krytycznym. Warstwa frontendu spełnia podstawową funkcjonalność potrzebną do funkcjonowania aplikacji.

Obawę o działanie aplikacji w krytycznych momentach budzą testy wydajnościowe dla kilku tysięcznej rzeszy użytkowników. Na ich podstawie zaleca się zmianę serwera publikującego aplikację lub zwiększenie przepustowości łączy. Dobrą praktyką jest również zmniejszenie zawartości graficznych elementów na stronie.

Aplikacja po zmianie serwera, szaty graficznej oraz zastosowaniu kilku innych rozwiązań działa lepiej, niż przed wykonaniem czynności z punktu 5. Największy wpływ na poprawę wydajności przyniósł MemCache, który pozwolił na przechowanie danych, dzięki czemu nie trzeba było ponownie odpytywać o nie serwera.

Zmiana wyglądu również dobrze wpłynęła na rozwój aplikacji, wygląda ona przejrzyściej i jest przyjazna dla użytkownika. Jednocześnie, skromność szaty graficznej może przyczynić się do pozytywnego przyjęcia strony przez użytkowników telefonów mobilnych.