Internetowy Serwis Aukcyjny

Jakub Niewczas, Damian Klimek, Tomasz Zabrzewski, Łukasz Myszkowski

Zespołowe przedsięwzięcie inżynierskie

Informatyka

Rok. akad. 2017/2018, sem. I

Prowadzący: dr hab. Marcin Mazur

Spis treści

1 Opis projektu

1.1 Członkowie zespołu

1. Damian Klimek (kierownik projektu);
2. Jakub Niewczas;
3. Tomasz Zabrzewski;
4. Łukasz Myszkowski;

1.2 Cel projektu (produkt)

Głównym i najważniejszym celem projektu jest stworzenie platformy webowej, która pojęcie „zakupy" pchnie krok dalej. Potencjalni klienci będą mieć możliwość robienia zakupów *(tj. sprzedawanie i kupowanie przedmiotów)* bez wychodzenia z domu. Kolejną ważną rzeczą jest zaimplementowanie modułu indywidualnych kont dla użytkowników *(logowanie się na swoje konto oraz rejestrowanie nowego)*. Również stawiamy nacisk na stworzenie miłego dla oka, prostego a przede wszystkim intuicyjnego designu strony internetowej aby użytkownicy mogli w łatwy sposób przemieszczać się po platformie i szybko wyszukiwać interesujące ich przedmioty.

1.3 Potencjalny odbiorca produktu (klient)

Konkretnym klientem może być każdy, kto potrzebuje pieniędzy przez co dostaje możliwość sprzedania czegokolwiek lub osoby potrzebujące jakiegoś dobra. Widełki wieku klientów nie są definiowalne - każdy kto posiada dostęp do internetu może skorzystać z usług jakie oferuje produkt.

1.4 Metodyka

Projekt będzie realizowany przy użyciu (zaadaptowanej do istniejących warunków) metodyki *Scrum*.

2 Wymagania użytkownika

2.1 User story 1

Jako nowy użytkownik serwisu chciałbym mieć możliwość założenia swojego indywidualnego konta, żebym mógł korzystać w pełni z usług dostarczanych przez serwis.

2.2 User story 2

Jako użytkownik chciałbym mieć możliwość zalogować się na swoje indywidualne konto, by móc korzystać z wszystkich usług platformy.

2.3 User story 3

Jako użytkownik chciałbym mieć możliwość wystawienia przedmiotu na aukcję, żebym mógł sprzedawać przedmioty oraz zarabiać pieniądze.

2.4 User story 4

Jako użytkownik podczas wystawiania przedmiotu na aukcję chciałbym mieć możliwość wybrania formy aukcji *(tj. licytacja, opcja „KUP TERAZ" lub obie formy)*, żebym mógł sprzedać przedmiot w wybranej formie aukcji.

2.5 User story 5

Jako użytkownik podczas wystawiania przedmiotu na aukcję chciałbym mieć możliwość wprowadzenia opisu aukcji:

- konkretna nazwa aukcji;
- dodatkowe informację o sprzedawanym przedmiocie;
- cena *(wywoławcza podczas licytacji lub stała za przedmiot podczas opcji „KUP TERAZ")*;
- wgranie zdjęcia przedmiotu;

żebym mógł zachęcić potencjalnego kupca do zakupu przedmiotu.

2.6 User story 6 (Opcjonalnie)

Jako użytkownik chciałbym mieć możliwość komentowania produktów innych osób wystawiających przedmioty na aukcji, , żebym mógł dowiedzieć się interesujących informacji od innych osób, które wcześniej kupiły przedmiot na jego temat bądź podzielić się własną opinią.

2.7 User story 7

Jako użytkownik chciałbym mieć możliwość dodania przedmiotów które mnie interesują do zakładek *(tj. Ulubione)*, żebym mógł szybko i łatwo je odnaleźć.

2.8 User story 8

Jako użytkownik chciałbym mieć możliwość dodania swoich danych:

- imię;
- nazwisko;
- miejscowość;
- numer telefonu;
- e-mail;
- numer konta bankowego;

żeby umożliwić innym użytkownikom kontakt ze mną i rozliczeń.

2.9 User story 9

Jako użytkownik chciałbym mieć możliwość zmiany:

- istniejącego hasła na nowe;
- istniejącego e-maila na nowy;
- danych osobowe na nowe;

żeby zaktualizować email oraz wprowadzić nowe hasło bądź też poprawić błąd w danych osobowych lub całkowicie je zmienić.

2.10 User story 10

Jako użytkownik chciałbym mieć możliwość zobaczenia ile razy odwiedzono moją aukcję *(tj. licznik odsłon)*, żebym mógł zobaczyć jakie jest zainteresowanie moją aukcją.

2.11 User story 11

Jako użytkownik chciałbym mieć możliwość edytowania danych aukcji *(tj. cena, opis)* oraz usunięcia danej aukcji, żebym mógł poprawić błedy lub zaktualizować dane aukcji.

2.12 User story 12

Jako użytkownik chciałbym mieć możliwość wyszukiwania interesujących mnie aukcji poprzez podanie słów kluczowych w wyszukiwarce, żebym mógł znaleźć przedmiot, który chciałbym zakupić.

2.13 User story 13

Jako użytkownik chciałbym by po dokonaniu transakcji (kup teraz bądź wygrana licytacja) wysyłała się do kupującego wiadomość z moim numerem konta oraz informacjami dotyczącymi transakcji, by mógł on dokonać wpłaty za zakupiony przedmiot.

2.14 User story 14

Jako użytkownik chciałbym mieć możliwość sprawdzenia starych aukcji które zostały zakończone (przedmiot sprzedano, przedmiot kupiono) bądź też wygasły, żebym mógl sprawdzić historie swoich transakcji.

2.15 User story 15

Jako użytkownik chciałbym mieć możliwość wysłania wiadomości prywatnej do innego użytkownika serwisu, żebym mógł się z nim skontaktować.

2.16 User story 16

Jako użytkownik chciałbym mieć możliwość przeglądania wszystkich dostępnych aukcji na serwisie w postaci listy wraz z ich podstawowymi danymi *(tj. nazwa aukcji, forma aukcji, cena, data rozpoczęcia oraz zakończenia aukcji)*, żebym mógł przeglądać aukcje oferowane przez serwis.

2.17 User story 17

Jako użytkownik chciałbym mieć możliwość zakupu produktu natychmiast (opcja kup teraz), bądź licytowania go (opcja licytuj), żebym mógł go kupić.

2.18 User story 18

Jako użytkownik chciałbym mieć możliwość posiadania paska nawigacyjnego w którym zostanie umiejscowiona wyszukiwarka, oraz panel konta użytkownika.

3 Harmonogram

3.1 Rejestr zadań (Product Backlog)

- Data rozpoczęcia: 24.10.2017;
- Data zakończenia: 14.11.2017;

3.2 Sprint 1

- Data rozpoczęcia: *31.10.2017*;
- Data zakończenia: *14.11.2017*;
- Scrum Master: *Łukasz Myszkowski*;
- Product Owner: *Tomasz Zabrzewski*;
- Development Team: *Jakub Niewczas, Damian Klimek, Łukasz Myszkowski*;

3.3 Sprint 2

- Data rozpoczęcia: *14.11.2017*;
- Data zakończenia: *28.11.2017*;
- Scrum Master: *Tomasz Zabrzewski*;
- Product Owner: *Damian Klimek*;
- Development Team: *Łukasz Myszkowski, Jakub Niewczas, Damian Klimek*;

3.4 Sprint 3

- Data rozpoczęcia: *28.11.2017*;
- Data zakończenia: *12.12.2017*;
- Scrum Master: *Damian Klimek*;
- Product Owner: *Jakub Niewczas*;
- Development Team: *Tomasz Zabrzewski, Łukasz Myszkowski, Jakub Niewczas*;

3.5 Sprint 4

- Data rozpoczęcia: *12.12.2017*;
- Data zakończenia: *09.01.2018*;
- Scrum Master: *Jakub Niewczas*;
- Product Owner: *Damian Klimek*;
- Development Team: *Tomasz Zabrzewski, Łukasz Myszkowski, Damian Klimek*;

3.6 Sprint 5

- Data rozpoczęcia: *09.01.2018*;
- Data zakończenia: *23.01.2018*;
- Scrum Master: *Łukasz Myszkowski*;
- Product Owner: *Tomasz Zabrzewski*;
- Development Team: *Damian Klimek, Jakub Niewczas, Tomasz Zabrzewski*;

4 Product Backlog

4.1 Backlog Item 1

Tytuł zadania: Baza danych

Opis zadania: Przygotowanie struktury bazy danych.

Priorytet: Wysoki

Definition of Done: Baza danych powinna zawierać tyle tabel ile potrzeba na stworzenie aplikacji. Środowisko bazodanowe **MySQL**. Baza danych serwisu musi posiadać następujące struktury.

Struktura tabeli **users**:

- **id** → [typ - `INT`, `AUTO_INCREMENT`];
- **username** → [typ - `VARCHAR(32)`, kodowanie - `UTF8_GENERAL_CI`];
- **password** → [typ - `VARCHAR(32)`, kodowanie - `UTF8_GENERAL_CI`];
- **firstname** → [typ - `VARCHAR(64)`, kodowanie - `UTF8_GENERAL_CI`];
- **surname** → [typ - `VARCHAR(64)`, kodowanie - `UTF8_GENERAL_CI`];
- **email** → [typ - `VARCHAR(64)`, kodowanie - `UTF8_GENERAL_CI`];
- **phone** → [typ - `VARCHAR(9)`, kodowanie - `UTF8_GENERAL_CI`];
- **place** → [typ - `VARCHAR(128)`, kodowanie - `UTF8_GENERAL_CI`];
- **bank** → [typ - `VARCHAR(26)`, kodowanie - `UTF8_GENERAL_CI`];
- **avatar** → [typ - `VARCHAR(128)`, kodowanie - `UTF8_GENERAL_CI`];

Struktura tabeli **categories**:

- **id** → [typ - `INT`, `AUTO_INCREMENT`];
- **name** → [typ - `VARCHAR(64)`, kodowanie - `UTF8_GENERAL_CI`];

Struktura tabeli **auctions**:

- **id** → [typ - `INT`, `AUTO_INCREMENT`];
- **name** → [typ - `VARCHAR(128)`, kodowanie - `UTF8_GENERAL_CI`];
- **category** → [typ - `INT`, relacja z kolumną id z tabeli categories];
- **buy_prize** → [typ - `VARCHAR(16)`, kodowanie - `UTF8_GENERAL_CI`];
- **bidding_prize** → [typ - `VARCHAR(16)`, kodowanie - `UTF8_GENERAL_CI`];
- **description** → [typ - `VARCHAR(512)`, kodowanie - `UTF8_GENERAL_CI`];
- **image** → [typ - `VARCHAR(256)`, kodowanie - `UTF8_GENERAL_CI`];
- **last_bidder** → [typ - `INT`, relacja z kolumną id z tabeli users];
- **buyer** → [typ - `INT`, relacja z kolumną id z tabeli users];

Struktura tabeli **comments**:

- **id** → [typ - `INT`, `AUTO_INCREMENT`];

- **message** → [typ - `VARCHAR(512)`, kodowanie - `UTF8_GENERAL_CI`];
- **owner** → [typ - `INT`, relacja z kolumną id z tabeli users];
- **auction** → [typ - `INT`, relacja z kolumną id z tabeli auctions];

Struktura tabeli **messages**:

- **id** → [typ - `INT`, `AUTO_INCREMENT`];
- **owner** → [typ - `INT`, relacja z kolumną id z tabeli users];
- **recipient** → [typ - `INT`, relacja z kolumną id z tabeli users];
- **title** → [typ - `VARCHAR(128)`, kodowanie - `UTF8_GENERAL_CI`];
- **message** → [typ - `VARCHAR(512)`, kodowanie - `UTF8_GENERAL_CI`];
- **readed** → [typ - `INT`];

4.2 Backlog Item 2

Tytuł zadania: Rejestracja

Opis zadania: Możliwość założenia nowego konta;

Priorytet: Wysoki

Definition of Done: Każda osoba powinna mieć możliwość założenia swojego indywidualnego konta. Należy stworzyć formularz z podstawowymi danymi (*tj. login, hasło, imię, nazwisko, e-mail, miejscowość, numer telefonu, numer konta bankowego oraz avatar*). Wpisane przez użytkownika dane muszą być pobierane metodą *POST*. Finalnym efektem rejestracji jest wysłanie zapytania do bazy danych, który utworzy nowy rekord w tabeli *users* z danymi użytkownika.

4.3 Backlog Item 3

Tytuł zadania: Zabezpieczenie hasła

Opis zadania: Wymuszenie od użytkownika wprowadzenia danych spełniających określone kryteria;

Priorytet: Niski

Definition of Done: Użytkownik tworzący nowe konto musi podać hasło składające się z minimum 8 znaków w tym:

- wielkie litery od A do Z;
- małe litery od a do z;
- cyfry od 0 do 9;
- znaki niealfabetyczne (np. `!, @, #, &`);

Należy stworzyć skrypt używający *wyrażenia regularne*, które w łatwy sposób mogą sprawdzić poprawność hasła z wyżej ustalonymi zasadami. Hasło, które będzie podawał użytkownik rejestrując lub logując się na swoje konto powinno być niewidoczne dla osób postronnych (wykropkowane).

4.4 Backlog Item 4

Tytuł zadania: Weryfikowanie danych

Opis zadania: Sprawdzanie wprowadzanych danych przez użytkownika podczas rejestracji;

Priorytet: Wysoki

Definition of Done: Każda osoba wprowadzająca dane do serwisu podczas rejestracji powinna podać poprawne dane:

- imię - powinno zawierać od 3 do 32 znaków, w tym tylko od a do z;
- nazwisko - powinno zawierać od 3 do 32 znaków, w tym tylko od a do z;
- e-mail - powinien przypominać maskę (tj. mójemail09@domena.pl);
- miejscowość - powinna zawierać od 3 do 64 znaków, w tym tylko od a do z;
- numer telefonu - powinna składać się z 9 cyfr.
- konto bankowe - powinno składać się z 26 cyfr.

Gdy użytkownik nie zastosuje się do wyżej wymienionych zasad, należy wysłać do niego w postaci *JavaScript Popup Alert box*, wiadomość zwrotną podającą, w którym miejscu popełnił błąd oraz jak poprawnie wypełnić input formularza.

4.5 Backlog Item 5

Tytuł zadania: Logowanie

Opis zadania: Możliwość zalogowania się na swoje konto;

Priorytet: Wysoki

Definition of Done: Stworzenie formularza logowania (tj. login oraz hasło) pobierające owe dane, następnie w postaci zapytania MySQL musi sprawdzać czy istnieją dane w bazie danych oraz czy pasują do jednego użytkownika. Należy zaimplementować wiadomość zwrotną gdy użytkownik wpisał złe dane (nie ma takiego rekordu w bazie danych bądź hasło nie pasuje do loginu). Po udanej operacji (logowanie powiodło się) należy stworzyć użytkownikowi sesję oraz przekierować go na stronę główną serwisu.

4.6 Backlog Item 6

Tytuł zadania: Pasek nawigacji

Opis zadania: Utworzenie paska nawigacji umieszczonego w górnej części ekranu;

Priorytet: Wysoki

Definition of Done: Należy utworzyć pasek nawigacji, który będzie umiejscowiony na górze ekranu przeglądarki. Pozycjonowanie paska należy ustawić jako *fixed*, aby w przypadku długiej strony „przykleił" się do ekranu i podążał za scrollowaną stroną. Szerokość paska powinna być uzależniona od szerokości ekranu, na którym będzie odwiedzany serwis, zaś wysokość powinna być ustawiona na *75px*.

4.7 Backlog Item 7

Tytuł zadania: Zmiana danych

Opis zadania: Umożliwienie użytkownikowi zmiany swoich podstawowych danych;

Priorytet: Średni

Definition of Done: Każda osoba posiadająca swoje konto powinna mieć możliwość zmiany swoich podstawowych danych na nowe (tj. imię, nazwisko, e-mail, miejscowość, numer telefonu, numer konta bankowego, hasło oraz avatar). Należy stworzyć formularz z wcześniej wymienionymi danymi oraz jako domyślne *value* kolejnych inputów formularza powinny być przypisane dane pobrane z bazy danych. Obowiązkowo należy zaimplementować przycisk *submit*, po kliknięciu którego zostaną pobrane dane z formularza metodą *POST* oraz wysłane do bazy danych w postaci zapytania MySQL. Wysłane dane muszą podmieniać już istniejące.

4.8 Backlog Item 8

Tytuł zadania: Wyszukiwarka

Opis zadania: Umożliwienie użytkownikowi wyszukiwania interesujących przedmiotów;

Priorytet: Średni

Definition of Done: Każda osoba (zalogowana lub niezalogowana) może używać wyszukiwarki. Wyszukiwarke w postaci formularza należy umieścić na pasku nawigacji, który znajduje się na górze ekranu. Wpisane hasło w input formularza zwrócone przez funkcję *htmlspecialchars()* należy umieścić w zapytaniu MySQL w poszukiwaniu przedmiotów. Zapytanie należy zaprojektować tak, aby wyszukiwał każdy przedmiot, w którym zawiera się wpisane przez użytkownika hasło *(np. dom, domek, domeczek, domuś itp.)*.

4.9 Backlog Item 9

Tytuł zadania: Panel konta

Opis zadania: Zgrupowanie w jedno miejsce podstawowych akcji użytkownika;

Priorytet: Średni

Definition of Done: Panel należy podzielić na dwie grupy - dla osób zalogowanych oraz dla osób niezalogowanych. Oba panele powinny być umiejscowione na pasku nawigacji na górze ekranu. Panel dla zalogowanego użytkownika powinien być w postaci avataru użytkownika, w który można kliknąć a następnie rozwija się menu z podstawowymi akcjami *(tj. edytuj dane, moje aukcje, historia zakupów, wyloguj)*. Panel dla osoby niezalogowanej powinien być w postaci napisu „Moje konto", które również rozwija się po kliknięciu w napis, z którego można wybrać dwie akcję *(tj. zarejestruj się, zaloguj się)*. Po kliknięciu, w któryś z odnośników z rozwijanych menu każdy powinien być przekierowywany na konkretne podstrony serwisu.

4.10 Backlog Item 10

Tytuł zadania: Dodanie aukcji

Opis zadania: Umożliwienie użytkownikowi wystawienia nowej aukcji;

Priorytet: Wysoki

Definition of Done: Każda osoba zalogowana na swoję konto powinna mieć tę opcję dostępną. Osoba, która nie jest zalogowana na swoje konto powinna dostać wiadomość zwrotną, że ta opcja jest tylko dla osób zalogowanych lub całkowicie zablokować dostęp. Tworzenie nowego rekordu w bazie danych w tabeli *auction*.

4.11 Backlog Item 11

Tytuł zadania: Formularz aukcyjny

Opis zadania: Umożliwienie użytkownikowi podanie niezbędnych danych dotyczących aukcji;

Priorytet: Wysoki

Definition of Done: Osoba zalogowana podczas wystawiania nowego przedmiotu na akcję musi mieć możliwość wpisania niezbędnych danych, które pozwolą odróżniać od siebie inne aukcje (tj. konkretna nazwa aukcji, opis przedmiotu, cena, zdjęcie). Formularz powinien pobierać dane metodą *POST*.

4.12 Backlog Item 12

Tytuł zadania: Edytowanie aukcji

Opis zadania: Umożliwienie użytkownikowi edytowania danych swojej aukcji;

Priorytet: Średni

Definition of Done: Użytkownik, który wystawił przedmiot na aukcję powinien mieć możliwość edytowania danych związanych z aukcją. Należy stworzyć formularz, w którym domyślnymi wartościami kolejnych inputów będą dane wczytane z bazy danych. Dane z formularza powinny być pobierane metodą *POST*. Kliknięcie przycisku *submit* skutować powinno nadpisanie istniejących danych w bazie poprzez wykonanie zapytania *UPDATE*.

4.13 Backlog Item 13

Tytuł zadania: Usuwanie aukcji

Opis zadania: Umożliwienie użytkownikowi usunięcia swojej aukcji;

Priorytet: Średni

Definition of Done: Użytkownik, który wystawił przedmiot na aukcję powinien mieć możliwość całkowitego usunięcia aukcji z serwisu. Usuwana aukcja powinna usunąć się również z bazy danych.

4.14 Backlog Item 14

Tytuł zadania: Lista aukcji

Opis zadania: Umożliwienie użytkownikowi wyświetlania dostępnych aukcji;

Priorytet: Wysoki

Definition of Done: Każda osoba odwiedzająca serwis powinna mieć możliwość zobaczenia wszystkich dostępnych aukcji. Na stronie głównej powinny być wczytane wszystkie aukcje z bazy danych, które są aktywne (nie przeminął czas zakończenia aukcji). Aukcje powinny być wyświetlane w postaci tabelki *(tj. miniaturka zdjęcia, nazwa aukcji, rodzaj aukcji oraz ceny)*. Nazwa aukcji powinna być odsyłaczem, który po kliknięciu przekieruje na podstrone przypisaną do aukcji.

4.15 Backlog Item 15

Tytuł zadania: Wiadomość informacyjna

Opis zadania: Wyświetlenie wiadomości informacyjnej dla niezalogowanych użytkowników;

Priorytet: Niski

Definition of Done: Należy stworzyć box, który będzie umieszczony na stronie głównej (zaraz pod paskiem nawigacji). Box powinien być widoczny tylko dla osób niezalogowanych oraz tylko na stronie głównej - po zalogowaniu całkowicie znika. Należy umieścić w nim wiadomość informacyjną odnośnie serwisu (co to za witryna) oraz powinno zawrzeć się odnośniki do zarejestrowania nowego konta oraz zalogowania.

4.16 Backlog Item 16

Tytuł zadania: Kupowanie przedmiotu

Opis zadania: Umożliwienie użytkownikowi zakupu;

Priorytet: Średni

Definition of Done: Osoba zalogowana powinna mieć możliwość zakupu interesującego przedmiotu poprzez opcję KUP TERAZ lub wygranie licytacji. Kiedy użytkownik zakupi przedmiot poprzez KUP TERAZ automatycznie dana aukcja powinna się zakończyć (wysyłane jest zapytanie do bazy danych edytujące rekord odpowiadający aukcji, do którego dopisywana jest wartość *id* osoby kupującej do kolumny *buyer*). Jeśli wartość jest 0, to znaczy, że nikt przedmiotu nie kupił. W przypadku wybrania formy aukcji jako licytacja osoba zalogowana może wpisać w formularzu cenę, którą jest w stanie zapłacić za przedmiot pod warunkiem, że jest ona większa od ostatniej licytacji. Formularz pobrany metodą *POST*. Zapytanie z nową zlicytowaną ceną nadpisuje się w bazie danych.

4.17 Backlog Item 17

Tytuł zadania: Informacja zwrotna

Opis zadania: Przesyłanie danych osób biorących udział w transakcji;

Priorytet: Wysoki

Definition of Done: Po sfinalizowaniu każdej transakcji powinny automatycznie przesyłać się dane, które będą pobierane z bazy danych:

- od kupującego do sprzedającego - imię, nazwisko, adres, numer telefonu;
- od sprzedającego do kupującego - imię, nazwisko, numer konta bankowego, numer transakcji, numer telefonu;

.

4.18 Backlog Item 18

Tytuł zadania: Historia zakupów

Opis zadania: Umożliwienie użytkownikowi przeglądania swoich starych transakcji;

Priorytet: Niski

Definition of Done: Każda osoba posiadająca swoje konto powinna mieć możliwość przeglądania swoich wszystkich wcześniejszych transakcji. Stworzyć podstronę przypisaną do każdego użytkownika serwisu, na którym będą wypisywane w postaci tabeli *(tj. miniaturka przedmiotu, nazwa, cena zakupu oraz jedna z kategorii [przedmiot: kupiony / sprzedany, aukcja wygasła])*. Inny użytkownik nie może podglądać historii zakupów innego użytkownika - ma dostęp tylko do swojej podstrony.

4.19 Backlog Item 19

Tytuł zadania: Wiadomości

Opis zadania: Umożliwienie użytkownikowi wysyłania i odbierania prywatnych wiadomości od innych użytkowników;

Priorytet: Średni

Definition of Done: Każda osoba posiadająca konto powinna mieć możlwiość wysyłania prywatnych wiadomości do innych użytkowników serwisu oraz również odbierania. Należy stworzyć formularz pozwalający wysyłanie wiadomości *(tj. temat wiadomości, do kogo ma być wysłana wiadomość oraz pole wiadomości)*. Formularz powinien być pobierany metodą *POST*. Pobrane dane powinny być umieszczone w bazie danych w tabeli *messages*. Domyślnie argument *readed* powinien być ustawiany na 0. W przypadku gdy *readed* ma wartość 0 wyświetlany jest na stronie głównej komunikat w postaci boxa pod paskiem nawigacji, informujący o nowej nieprzeczytanej prywatnej wiadomości. Jeśli użytkownik wejdzie w wiadomość automatycznie wartość argumentu *readed* zmienia się na 1, zapisuje się w bazie - przez to nie będzie wyświetlane powiadomienie

o nieprzeczytanej wiadomości. Należy stworzyć podstronę, w której będą wyświetlać się wszystkie prywatne wiadomości użytkownika w postaci odsyłacza, który po kliknięciu będzie przekierowywał na kolejną podstronę przypisaną do konkretnej wiadomości. Na podstronie musi znajdować się podgląd wiadomości *(tj. od kogo została przysłana, temat wiadomości oraz cała wiadomość)*. Dane wczytane bezpośrednio z bazy danych.

4.20 Backlog Item 20

Tytuł zadania: Komentarze

Opis zadania: Umożliwienie użytkownikowi komentowania konkretnych aukcji;

Priorytet: Niski

Definition of Done: Każda osoba zalogowana na swoje konto powinna mieć możliwość skorzystania z opcji komentowania aukcji. Należy stworzyć formularz tylko dla zalogowanych osób pod aukcją *(tj. wiadomość)*. Dane pobierane poprzez metodę *POST*. Dane powinny być wysyłane do bazy danych wraz z *id* konta, z którego pisany jest komentarz. Pod formularzem należy stworzyć podgląd komentarzy do aukcji. W przypadku gdy aukcja nie posiada żadnych komentarzy należy wyświetlić wiadomość informującą o tym stanie. W przeciwnym wypadku należy wyświetlić w postaci tabeli komentarz *(tj. login użytkownika, który napisał komentarz, avatar oraz wiadomość komentarza)*. Osoby niezalogowane pod aukcją widzą tylko komentarze, formularz jest dla nich całkowicie niewidoczny.

4.21 Backlog Item 21

Tytuł zadania: Ulubione

Opis zadania: Umożliwienie użytkownikowi zapisania aukcji;

Priorytet: Niski

Definition of Done: Każda osoba zalogowana na swoje konto powinna mieć możliwość przypisania linku interesującej go aukcji do odpowiedniej zakładki w swoim profilu.

4.22 Backlog Item 22

Tytuł zadania: Licznik

Opis zadania: Umożliwienie użytkownikowi sprawdzenia ilości odsłon aukcji;

Priorytet: Niski

Definition of Done: Na każdej stronie aukcji w dogodnym miejscu pod ogłoszeniem powinien znajdować się licznik odwiedzin danej aukcji. Licznik nalicza się gdy osoba wejdzie na podstronę aukcji. Dana powinna być zapisywana i wczytywana z bazy danych.

4.23 Backlog Item 23

Tytuł zadania: Kategorie

Opis zadania: Umożliwienie użytkownikowi wyszukiwania aukcji po kategoriach;

Priorytet: Niski

Definition of Done: Każda osoba odwiedzająca serwis powinna mieć dostępny pasek boczny z dostępnymi kategoriami. Kategorie powinny być wczytywane z bazy danych z tabeli *categories*, w postaci odsyłacza który umożliwi przekierowanie osoby na podstronę wybranej kategorii, gdzie zostaną wyświetlone wszystkie aukcje należące do danej kategorii.

5 Sprint 1

5.1 Cel

Celem pierwszego Sprintu jest umożliwienie użytkownikowi założenia konta dzięki któremu będzie on mógł uzyskać dostęp do wszystkich opcji serwisu i korzystania z niego. Aby to mogło się odbyć wcześniej zostanie stworzona baza danych w której będą zapisywane wszystkie dane.

5.2 Sprint Planning/Backlog

Tytuł zadania. Baza danych.

- Estymata: XL

Tytuł zadania. Rejestracja

- Estymata: S

Tytuł zadania. Logowanie

- Estymata: S

Tytuł zadania. Weryfikowanie danych

- Estymata: S

«Tutaj dodawać kolejne zadania»

5.3 Realizacja

Tytuł zadania. Logowanie

Wykonawca. Damian Klimek

Realizacja. Został stworzony formularz logowania który wymaga od użytkownika podania danych *(tj. login oraz hasło)*, następnie pobiera dane z formularza metodą *POST* i wartości przypisuje do zmiennych, które później sprawdzane są w bazie danych za pomocą zapytania MySQL. Jeśli nie istnieją takie dane to użytkownik otrzymuje informacje zwrotną w postaci wyskakującego okienka, że konto nie istnieje, a w przeciwnym razie zostaje utworzona sesja oraz zostaje zalogowany.

Kod programu:

```
if($_GET['action'] === "login")
{
if(!empty($_SESSION['username']) || !empty($_SESSION['password']))
{
header("Refresh:0; url=index.php");
echo "<script language='javascript'>alert('Jesteś już zalogowany!');</script>";
return 0;
}

if(isset($_POST['loginSubmit']))
{
$username = $_POST['loginName'];
$password = $_POST['loginPassword'];

$query = mysql_query("SELECT * FROM users WHERE username LIKE '".$username."' AND password
//$userLogged = mysql_fetch_array($query);

if(mysql_num_rows($query) > 0)
{
$_SESSION['password'] = $password;
$_SESSION['username'] = $username;
header("Refresh:0; url=index.php");
}
else
{
header("Refresh:0; url=index.php?action=login");
echo "<script language='javascript'>alert('Podałeś błędne dane!');</script>";
}
}

$index = '
<div class="container" style="margin-top: 20px;">
<div class="row">
<div class="backgroundBox">
<form class="form-horizontal" action="" method="post">
```

```
<div class="form-group">
<label for="userName" class="col-sm-2 control-label">Nazwa użytkownika</label>
<div class="col-sm-10">
<input type="text" class="form-control input-lg" id="userName" name="loginName" placeholde
</div>
</div>
<div class="form-group">
<label for="userPassword" class="col-sm-2 control-label">Hasło</label>
<div class="col-sm-10">
<input type="password" class="form-control input-lg" id="userPassword" name="loginPassword
</div>
</div>
<div class="form-group">
<div class="col-sm-offset-2 col-sm-10">
<button type="submit" name="loginSubmit" class="btn btn-default input-lg"><span style="fon
</div>
</div>
</form>
</div>
</div>
</div>
';
}
```

Tytuł zadania. Baza danych;

Wykonawca. Kuba Niewczas;

Realizacja. Baza danych została stworzona w środowisku *MySQL*. Struktura bazy danych tworzona graficznie za pomocą *PHPMyAdmin*. Wszystkie dane będą umieszczane w bazie o nazwie *zpi_project*, domyślne porównywanie napisów (kodowanie) zostało ustawione na `UTF8_GENERAL_CI`. Baza danych zawiera następujące tabele *(tj. auctions, categories, comments, messages, users)*. Wykonując to zadanie nie napotkałem żadnych problemów lecz było dość czasochłonne ponieważ trzeba było przemyśleć całą strukturę działania serwisu.

Struktura tabeli *auctions*:

```
--
-- Struktura tabeli dla tabeli ‘auctions‘
--

CREATE TABLE IF NOT EXISTS ‘auctions‘ (
‘id‘ int(11) NOT NULL AUTO_INCREMENT,
‘name‘ varchar(128) NOT NULL,
‘category‘ int(11) NOT NULL,
‘buy_prize‘ varchar(16) NOT NULL,
‘bidding_prize‘ varchar(16) NOT NULL,
```

	#	Nazwa	Typ	Metoda porównywania napisów	Atrybuty	Null	Ustawienia domyślne	Dodatkowo
☐	1	id	int(11)			Nie	*Brak*	AUTO_INCREMENT
☐	2	name	varchar(128)	utf8_general_ci		Nie	*Brak*	
☐	3	category	int(11)			Nie	*Brak*	
☐	4	buy_prize	varchar(16)	utf8_general_ci		Nie	*Brak*	
☐	5	bidding_prize	varchar(16)	utf8_general_ci		Nie	*Brak*	
☐	6	description	varchar(512)	utf8_general_ci		Nie	*Brak*	
☐	7	image	varchar(256)	utf8_general_ci		Nie	*Brak*	
☐	8	last_bidder	int(11)			Nie	*Brak*	
☐	9	buyer	int(11)			Nie	*Brak*	

Rysunek 1: Tabela auctions

```
‘description‘ varchar(512) NOT NULL,
‘image‘ varchar(256) NOT NULL,
‘last_bidder‘ int(11) NOT NULL,
‘buyer‘ int(11) NOT NULL,
PRIMARY KEY (‘id‘)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT=’Aukcje’ AUTO_INCREMENT=1 ;
```

Struktura tabeli *categories*:

	#	Nazwa	Typ	Metoda porównywania napisów	Atrybuty	Null	Ustawienia domyślne	Dodatkowo
☐	1	id	int(11)			Nie	*Brak*	AUTO_INCREMENT
☐	2	name	varchar(64)	utf8_general_ci		Nie	*Brak*	

Rysunek 2: Tabela categories

```
--
-- Struktura tabeli dla tabeli ‘categories‘
--

CREATE TABLE IF NOT EXISTS ‘categories‘ (
‘id‘ int(11) NOT NULL AUTO_INCREMENT,
‘name‘ varchar(64) NOT NULL,
PRIMARY KEY (‘id‘)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COMMENT=’Kategorie aukcji’ AUTO_INCREMENT=1 ;
```

Struktura tabeli *comments*:

```
--
-- Struktura tabeli dla tabeli ‘comments‘
--

CREATE TABLE IF NOT EXISTS ‘comments‘ (
‘id‘ int(11) NOT NULL AUTO_INCREMENT,
‘message‘ varchar(512) NOT NULL,
```

	#	Nazwa	Typ	Metoda porównywania napisów	Atrybuty	Null	Ustawienia domyślne	Dodatkowo
☐	1	**id**	int(11)			Nie	*Brak*	AUTO_INCREMENT
☐	2	**message**	varchar(512)	utf8_general_ci		Nie	*Brak*	
☐	3	**owner**	int(11)			Nie	*Brak*	
☐	4	**auction**	int(11)			Nie	*Brak*	

Rysunek 3: Tabela comments

```
‘owner‘ int(11) NOT NULL,
‘auction‘ int(11) NOT NULL,
PRIMARY KEY (‘id‘)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT=’Komentarze do aukcji’ AUTO_INCREMENT=1 ;
```

Struktura tabeli *messages*:

	#	Nazwa	Typ	Metoda porównywania napisów	Atrybuty	Null	Ustawienia domyślne	Dodatkowo
☐	1	**id**	int(11)			Nie	*Brak*	AUTO_INCREMENT
☐	2	**owner**	int(11)			Nie	*Brak*	
☐	3	**recipient**	int(11)			Nie	*Brak*	
☐	4	**title**	varchar(128)	utf8_general_ci		Nie	*Brak*	
☐	5	**message**	varchar(512)	utf8_general_ci		Nie	*Brak*	
☐	6	**readed**	int(1)			Nie	*Brak*	

Rysunek 4: Tabela messages

```
CREATE TABLE IF NOT EXISTS ‘messages‘ (
‘id‘ int(11) NOT NULL AUTO_INCREMENT,
‘owner‘ int(11) NOT NULL,
‘recipient‘ int(11) NOT NULL,
‘title‘ varchar(128) NOT NULL,
‘message‘ varchar(512) NOT NULL,
‘readed‘ int(1) NOT NULL,
PRIMARY KEY (‘id‘)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT=’Prywatne wiadomości’ AUTO_INCREMENT=1 ;
```

Struktura tabeli *users*:

```
CREATE TABLE IF NOT EXISTS ‘users‘ (
‘id‘ int(11) NOT NULL AUTO_INCREMENT,
‘username‘ varchar(32) NOT NULL,
‘password‘ varchar(32) NOT NULL,
‘firstname‘ varchar(64) NOT NULL,
‘surname‘ varchar(64) NOT NULL,
‘email‘ varchar(64) NOT NULL,
‘phone‘ varchar(9) NOT NULL,
‘place‘ varchar(128) NOT NULL,
‘bank‘ varchar(26) NOT NULL,
‘avatar‘ varchar(128) NOT NULL,
```

	#	Nazwa	Typ	Metoda porównywania napisów	Atrybuty	Null	Ustawienia domyślne	Dodatkowo
☐	1	id	int(11)			Nie	*Brak*	AUTO_INCREMENT
☐	2	username	varchar(32)	utf8_general_ci		Nie	*Brak*	
☐	3	password	varchar(32)	utf8_general_ci		Nie	*Brak*	
☐	4	firstname	varchar(64)	utf8_general_ci		Nie	*Brak*	
☐	5	surname	varchar(64)	utf8_general_ci		Nie	*Brak*	
☐	6	email	varchar(64)	utf8_general_ci		Nie	*Brak*	
☐	7	phone	varchar(9)	utf8_general_ci		Nie	*Brak*	
☐	8	place	varchar(128)	utf8_general_ci		Nie	*Brak*	
☐	9	bank	varchar(26)	utf8_general_ci		Nie	*Brak*	
☐	10	avatar	varchar(128)	utf8_general_ci		Nie	*Brak*	

Rysunek 5: Tabela users

```
PRIMARY KEY (‘id‘)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COMMENT=’Konta użytkowników serwisu’ AUTO_INCREMENT=
```

Tytuł zadania. Rejestracja;

Wykonawca. Kuba Niewczas;

Realizacja. Tworząc skrypt rejestacji przypisałem pobrane dane z formularza metodą *POST* do zmiennych. W postaci zapytania *MySQL* dane zawarte w zmiennych będą dodawane do bazy danych do tabeli *users*. Zadanie nie wymagało zbyt wielkiego poświęcenia czasu gdyż składa się tylko z paru operacji.

```
else if($_GET[’action’] === "register")
{
if(isset($_POST[’registerSubmit’]))
{
$username = $_POST[’registerName’];
$password = $_POST[’registerPassword’];
$firstname = $_POST[’registerFirstname’];
$surname = $_POST[’registerSurname’];
$email = $_POST[’registerEmail’];
$place = $_POST[’registerPlace’];
$phone = $_POST[’registerPhone’];
$bank = $_POST[’registerBank’];

mysql_query("INSERT INTO users (username, password, firstname, surname, email, phone, plac
header("Refresh:0; url=index.php?action=login");
echo "<script language=’javascript’>alert(’Pomyślnie założyłeś konto - możesz teraz się za
}

$index = ’
<div class="container" style="margin-top: 20px;">
<div class="row">
<div class="backgroundBox">
```

```
<form class="form-horizontal" action="" method="post">
<div class="form-group">
<label for="userName" class="col-sm-2 control-label">Nazwa użytkownika</label>
<div class="col-sm-10">
<input type="text" class="form-control input-lg" id="userName" name="registerName" placeho
</div>
</div>
<div class="form-group">
<label for="userPassword" class="col-sm-2 control-label">Hasło</label>
<div class="col-sm-10">
<input type="password" class="form-control input-lg" id="userPassword" name="registerPassw
</div>
</div>
<div class="form-group">
<label for="userFirstname" class="col-sm-2 control-label">Imię</label>
<div class="col-sm-10">
<input type="text" class="form-control input-lg" id="userFirstname" name="registerFirstnam
</div>
</div>
<div class="form-group">
<label for="userSurname" class="col-sm-2 control-label">Nazwisko</label>
<div class="col-sm-10">
<input type="text" class="form-control input-lg" id="userSurname" name="registerSurname" p
</div>
</div>
<div class="form-group">
<label for="userMail" class="col-sm-2 control-label">E-mail</label>
<div class="col-sm-10">
<input type="email" class="form-control input-lg" id="userMail" name="registerEmail" place
</div>
</div>
<div class="form-group">
<label for="userPlace" class="col-sm-2 control-label">Miejscowość</label>
<div class="col-sm-10">
<input type="text" class="form-control input-lg" id="userPlace" name="registerPlace" place
</div>
</div>
<div class="form-group">
<label for="userPhone" class="col-sm-2 control-label">Telefon</label>
<div class="col-sm-10">
<input type="text" class="form-control input-lg" id="userPhone" name="registerPhone" place
</div>
</div>
<div class="form-group">
<label for="userBank" class="col-sm-2 control-label">Konto bankowe</label>
<div class="col-sm-10">
<input type="text" class="form-control input-lg" id="userBank" name="registerBank" placeho
</div>
</div>
<div class="form-group">
```

```
<div class="col-sm-offset-2 col-sm-10">
<button type="submit" name="registerSubmit" class="btn btn-default input-lg"><span style="
</div>
</div>
</form>
</div>
</div>
</div>
';
}
```

Tytuł zadania. Weryfikacja danych;

Wykonawca. Kuba Niewczas;

Realizacja. Do wcześniejszego zadania **Rejestracja** dodałem warunki, które muszą być spełnione aby użytkownik mógł założyć swoje konto. Użyłem zmiennych, które zostały zadeklarwane w bazowym zadaniu. Na początku wykonałem dwa zapytania do bazy danych aby sprawdzić czy wpisany *login oraz hasło* istnieją już w bazie - jeśli tak to zwracana jest wiadomość, że konto na takie dane już istnieje. Lista warunków z backlogu została oskryptowana, w przypadku gdy skrypt ustali, że użytkownik popełnił błąd zostanie mu zwrócona informacja w postaci wyskakującego okienka *JavaScriptu*.

```
$query = mysql_query("SELECT NULL FROM users WHERE username LIKE '".$username."'");
$query2 = mysql_query("SELECT NULL FROM users WHERE email LIKE '".$email."'");

if(mysql_num_rows($query) > 0)
{
echo "<script language='javascript'>alert('Taka nazwa użytkownika jest zajęta - wybierz in
}
else if(mysql_num_rows($query2) > 0)
{
echo "<script language='javascript'>alert('Taki adres e-mail jest już zarejestrowany - wyb
}
else if(strlen($username) > 32 || strlen($username) < 3)
{
echo "<script language='javascript'>alert('Nazwa użytkownika powinna zawierać 3 - 32 znakó
}
else if(strlen($password) > 32 || strlen($password) < 3)
{
echo "<script language='javascript'>alert('Hasło powinno zawierać 3 - 32 znaków!');</scrip
}
else if(strlen($firstname) > 64 || strlen($firstname) < 3)
{
echo "<script language='javascript'>alert('Imię powinno zawierać 3 - 64 znaków!');</script
}
else if(preg_match('/[0-9]+/g', $firstname))
{
```

```
echo "<script language='javascript'>alert('Imię powinno zawierać tylko litery!');</script>
}
else if(strlen($surname) > 64 || strlen($surname) < 3)
{
echo "<script language='javascript'>alert('Nazwisko powinno zawierać 3 - 64 znaków!');</sc
}
else if(preg_match('/[0-9]+/g', $surname))
{
echo "<script language='javascript'>alert('Nazwisko powinno zawierać tylko litery!');</scr
}
else if(strlen($place) > 128 || strlen($place) < 3)
{
echo "<script language='javascript'>alert('Miejscowość powinna zawierać 3 - 128 znaków!');
}
else if(strlen($phone) != 9)
{
echo "<script language='javascript'>alert('Numer telefonu składa się z 9 cyfr!');</script>
}
else if(!is_numeric($phone))
{
echo "<script language='javascript'>alert('Numer telefonu składa się z samych cyfr!');</sc
}
else if(strlen($bank) != 26)
{
echo "<script language='javascript'>alert('Numer konta bankowego składa się z 26 cyfr!');<
}
else if(!is_numeric($bank))
{
echo "<script language='javascript'>alert('Numer konta bankowego składa się z samych cyfr!
}
else
{
mysql_query("INSERT INTO users (username, password, firstname, surname, email, phone, plac
header("Refresh:0; url=index.php?action=login");
echo "<script language='javascript'>alert('Pomyślnie założyłeś konto - możesz teraz się za
}
```

5.4 Sprint Review/Demo

«Sprawozdanie z przeglądu Sprint'u – czy założony cel (przyrost) został osiągnięty oraz czy wszystkie zaplanowane Backlog Item'y zostały zrealizowane? Demostracja przyrostu produktu».

6 Sprint 2

6.1 Cel

«Określić, w jakim celu tworzony jest przyrost produktu».

6.2 Sprint Planning/Backlog

Tytuł zadania. «Tytuł».

- Estymata: «szacowana czasochłonność (w „koszulkach")».

Tytuł zadania. «Tytuł».

- Estymata: «szacowana czasochłonność (w „koszulkach")».

«Tutaj dodawać kolejne zadania»

6.3 Realizacja

Tytuł zadania. «Tytuł».

Wykonawca. «Wykonawca».

Realizacja. «Sprawozdanie z realizacji zadania (w tym ocena zgodności z estymatą). Kod programu (środowisko `verbatim`):

```
for (i=1; i<10; i++)
...
```

».

Tytuł zadania. «Tytuł».

Wykonawca. «Wykonawca».

Realizacja. «Sprawozdanie z realizacji zadania (w tym ocena zgodności z estymatą). Kod programu (środowisko `verbatim`):

```
for (i=1; i<10; i++)
...
```

».

«Tutaj dodawać kolejne zadania»

6.4 Sprint Review/Demo

«Sprawozdanie z przeglądu Sprint'u – czy założony cel (przyrost) został osiągnięty oraz czy wszystkie zaplanowane Backlog Item'y zostały zrealizowane? Demostracja przyrostu produktu».

«Tutaj dodawać kolejne Sprint'y»

Literatura

[1] S. R. Covey, *7 nawyków skutecznego działania*, Rebis, Poznań, 2007.

[2] Tobias Oetiker i wsp., Nie za krótkie wprowadzenie do systemu LaTeX 2_ε, `ftp://ftp.gust.org.pl/TeX/info/lshort/polish/lshort2e.pdf`

[3] K. Schwaber, J. Sutherland, *Scrum Guide*, `http://www.scrumguides.org/`, 2016.

[4] `https://agilepainrelief.com/notesfromatooluser/tag/scrum-by-example`

[5] `https://www.tutorialspoint.com/scrum/scrum_user_stories.htm`