Aplikacja "Memorize!"

Damian Hadała, Dawid Sułowicz, Krzysztof Radoń, Krzysztof Jamro

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. Krzysztof Radoń (kierownik projektu).
2. Damian Hadała.
3. Dawid Sułowicz.
4. Krzysztof Jamro.

1.2 Cel projektu (produkt)

Aplikacja internetowa "Memorize!" pozwalająca użytkownikom zapisywać interesujące ich informacje i tym samym dodawać wpisy do dziennika podczas poznawania danego zagadnienia. Po zalogowaniu się użytkownik będzie miał możliwość utworzenia nowego tematu, dodania nowego wpisu, a także odczytywania i edytowania istniejących wpisów.

1.3 Potencjalny odbiorca produktu (klient)

Użytkownicy poznający nowe zagadnienia prowadzą dziennik, w którym zapisują to, czego się nauczyli. Dzięki aplikacji "Memorize!" proces nauki i zapamiętywania może być znacznie efektywniejszy.

1.4 Metodyka

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

2 Wymagania użytkownika

2.1 User story 1

Jako użytkownik aplikacji "Memorize!" chcę mieć możliwość dostępu do listy tematów, aby móc je w dowolnej chwili podglądać i sprawdzać ich zawartość.

2.2 User story 2

Jako użytkownik aplikacji "Memorize!" chcę mieć możliwość tworzenia nowych tematów, abym później mógł do nich dodawać powiązane wpisy.

2.3 User story 3

Jako użytkownik aplikacji "Memorize!" chcę mieć możliwość sprawdzenia daty i godziny umieszczenia każdego wpisu, aby móc weryfikować poczynione przeze mnie postępy w nauce.

2.4 User story 4

Jako użytkownik aplikacji "Memorize!" chcę by długie wpisy nie były wyświetlane w postaci ciągłego strumienia tekstu, aby poprawić czytelność podczas ich przeglądania.

2.5 User story 5

Jako użytkownik aplikacji "Memorize!" chcę żeby moje wpisy były prywatne tak, abym tylko ja mógł nimi zarządzać.

2.6 User story 6

Jako użytkownik aplikacji "Memorize!" chcę mieć możliwość edytowania wpisów, abym mógł zmieniać ich treść w pożądanej przeze mnie chwili.

2.7 User story 7

Jako niezalogowany użytkownik aplikacji "Memorize!" chcę mieć możliwość zarejestrowania własnego profilu, abym mógł dodawać własne treści.

2.8 User story 8

Jako użytkownik aplikacji "Memorize!" chcę żeby moje konto było chronione hasłem tak, aby inny użytkownik nie uzyskał do niego dostępu.

2.9 User story 9

Jako niezalogowany użytkownik aplikacji "Memorize!" chcę zobaczyć instrukcję wyjaśniającą przeznaczenie tej aplikacji, abym mógł zapoznać się z jej działaniem.

2.10 User story 10

Jako użytkownik aplikacji "Memorize!" chcę mieć dostęp z każdego komputera do listy selektorów i reguł, których cały czas się uczę.

2.11 User story 11

Jako użytkownik aplikacji "Memorize!" chcę mieć dostęp z każdego komputera do listy tematów prowadzonych przeze mnie zajęć, abym mógł się w nich zorientować pracując w kilku placówkach edukacyjnych.

2.12 User story 12

Jako użytkownik aplikacji "Memorize!" chcę mieć dostęp z każdego komputera do listy moich inspiracji, abym nie musiał szukać ich papierowej wersji na wypadek, gdybym ją zgubił.

2.13 User story 13 (optional)

Jako użytkownik aplikacji "Memorize!" chcę mieć dostęp do aplikacji z urządzenia mobilnego, aby móc z niej swobodnie korzystać z dala od komputera.

2.14 User story 14 (optional)

Jako użytkownik aplikacji "Memorize!" chcę mieć możliwość dodawania do wpisów załączników, abym mógł przechowywać w nich robocze wersje dokumentów.

2.15 User story 15 (optional)

Jako użytkownik aplikacji "Memorize!" chcę mieć możliwość formatowania tekstu wpisu, aby móc wyszczególnić ważniejsze jego elementy.

3 Harmonogram

3.1 Rejestr zadań (Product Backlog)

- Data rozpoczęcia: 24.10.2017 r.
- Data zakończenia: 7.11.2017 r.

3.2 Sprint 1

- Data rozpoczęcia: 7.11.2017 r.
- Data zakończenia: 28.11.2017 r.
- Scrum Master: Damian Hadała.
- Product Owner: Krzysztof Radoń.
- Development Team: Krzysztof Radoń, Dawid Sułowicz, Krzysztof Jamro, Damian Hadała.

3.3 Sprint 2

- Data rozpoczęcia: 28.11.2017 r.
- Data zakończenia: 12.12.2017 r.
- Scrum Master: Krzysztof Jamro.
- Product Owner: Damian Hadała.
- Development Team: Krzysztof Radoń, Dawid Sułowicz, Krzysztof Jamro, Damian Hadała.

3.4 Sprint 3

- Data rozpoczęcia: 12.12.2017 r.
- Data zakończenia: 09.01.2018 r.
- Scrum Master: Krzysztof Radoń.
- Product Owner: Dawid Sułowicz.
- Development Team: Krzysztof Radoń, Dawid Sułowicz, Krzysztof Jamro, Damian Hadała.

3.5 Sprint 4

- Data rozpoczęcia: 09.01.2018 r.
- Data zakończenia: 23.01.2018 r.
- Scrum Master: Dawid Sułowicz.
- Product Owner: Krzysztof Jamro.
- Development Team: Krzysztof Radoń, Dawid Sułowicz, Krzysztof Jamro, Damian Hadała.

4 Product Backlog

4.1 Backlog Item 1

Tytuł zadania. Instalacja i konfiguracja Django.

Opis zadania. Django to darmowy i open - source'owy framework, którego instalacja umożliwi nam utworzenie naszej aplikacji internetowej. Poprzez użycie poleceń niewielkiego programu, jakim jest plik manage.py uzyskamy możliwość zarządzania pracą serwera oraz bazy danych. Natomiast plik settings.py będzie kontrolował współdziałanie Django z systemem operacyjnym, a także da nam możliwość zarządzania projektem.

Priorytet. Wysoki.

Definition of Done. Skonfigurowany projekt wstępny oraz serwer.

4.2 Backlog Item 2

Tytuł zadania. Utworzenie bazy danych.

Opis zadania. Tworzymy tabele bazy danych w celu przechowywania informacji używanych w projekcie. Większość informacji związanych z projektem jest przez Django przechowywana w bazie danych.

Priorytet. Wysoki.

Definition of Done. Skonfigurowany framework pracujący z bazą danych.

4.3 Backlog Item 3

Tytuł zadania. Utworzenie i konfiguracja konta super - użytkownika.

Opis zadania. Utworzenie konta superużytkownika w celu nadania administratorowi możliwości dostępu do wszystkich przechowywanych informacji naszej aplikacji sieciowej.

Priorytet. Wysoki.

Definition of Done. Bezpieczny dostęp do panelu administracyjnego dla właściciela aplikacji.

4.4 Backlog Item 4

Tytuł zadania. Zdefiniowanie odpowiednich klas.

Opis zadania. Tworzymy klasę Topic zawierającą tylko dwa atrybuty: text i dateAdded, dzięki której użytkownik uzyska możliwość tworzenia wielu różnych tematów oraz sprawdzi kiedy dany wpis został dodany. Utworzymy też klasę Entry, która usprawni możliwość dodawania wpisów poprzez powiązanie ich z określonym tematem.

Priorytet. Wysoki.

Definition of Done. Możliwość dodawania wpisów. Każdy wpis jest powiązany z określonym tematem.

4.5 Backlog Item 5

Tytuł zadania. Ograniczenie długości wyświetlanych wpisów na stronie tematu.

Opis zadania. Ponieważ cały wpis może mieć postać dość długiego tekstu, nakazujemy wyświetlenie jedynie pierwszych 50 znaków przechowywanych przez atrybut text. Dołączamy również wielokropek, aby wyraźnie wskazać, ze nie został wyświetlony pełny wpis.

Priorytet. Wysoki.

Definition of Done. Długość wyświetlanych wpisów na stronie tematu ograniczona do pierwszych 50 znaków.

4.6 Backlog Item 6

Tytuł zadania. Utworzenie strony głównej naszej aplikacji internetowej.

Opis zadania. Przygotowanie adresu URL strony głównej, który będzie bazowym adresem URL podawanym przez użytkownika w celu uzyskania dostępu do aplikacji. Nadajemy wygląd naszej stronie głównie poprzez dodanie funkcji widoku i utworzenie potrzebnych szablonów w celu spełnienia oczekiwać użytkownika.

Priorytet. Wysoki.

Definition of Done. Możliwość wyświetlenia strony głównej naszej aplikacji.

4.7 Backlog Item 7

Tytuł zadania. Utworzenie stron wyświetlających tematy i zawarte w nich wpisy.

Opis zadania. Utworzenie strony internetowej poprzez zdefiniowanie jej adresu URL, która nada użytkownikowi możliwość wglądu w tematy oraz poszczególne wpisy w tematach, a także poinformuje o braku tematów lub wpisów w poszczególnych tematach.

Priorytet. Wysoki.

Definition of Done. Możliwość wyświetlania tematów oraz powiązanych z nimi wpisów.

4.8 Backlog Item 8

Tytuł zadania. Utworzenie łącza ze strony tematów.

Opis zadania. Zmodyfikowanie szablonu tematów tak, aby każde łącze tematu prowadziło na odpowiednią stronę w celu przejrzystego przeglądania tematów przez użytkownika.

Priorytet. Średni.

Definition of Done. Możliwość przeglądania tematów na oddzielnych stronach.

4.9 Backlog Item 9

Tytuł zadania. Dodawanie nowych tematów.

Opis zadania. Użycie oferowanych przez Django narzędzi do budowania formularzy, aby przygotować stronę pozwalającą każdemu użytkownikowi na dodawanie nowych tematów.

Priorytet. Wysoki.

Definition of Done. Możliwość dodawania nowych tematów przez wszystkich użytkowników.

4.10 Backlog Item 10

Tytuł zadania. Utworzenie przekierowania do listy tematów po dodaniu nowego tematu.

Opis zadania. Konieczne jest przeniesienie użytkownika z powrotem na stronę z tematami.

Priorytet. Średni.

Definition of Done. Przekierowanie użytkownika po dodaniu nowego tematu na stronę z tematami.

4.11 Backlog Item 11

Tytuł zadania. Utworzenie przycisku dodającego nowy temat.

Opis zadania. Django nie tworzy przycisku wysyłającego formularz, więc definiujemy go sami.

Priorytet. Wysoki.

Definition of Done. Przycisk dodający nowy temat.

4.12 Backlog Item 12

Tytuł zadania. Dodawanie nowych wpisów.

Opis zadania. Użycie oferowanych przez Django narzędzi do budowania formularzy, aby przygotować stronę pozwalającą każdemu użytkownikowi na dodawanie nowych wpisów do tematów.

Priorytet. Wysoki.

Definition of Done. Możliwość dodawania nowych wpisów do tematów przez wszystkich użytkowników.

4.13 Backlog Item 13

Tytuł zadania. Utworzenie przekierowania na stronę tematu po dodaniu nowego wpisu.

Opis zadania. Po umieszczeniu nowego wpisu użytkownik powinien zobaczyć wpis na liście wszystkich wpisów dla tematu.

Priorytet. Średni.

Definition of Done. Przekierowanie użytkownika po dodaniu nowego wpisu na stronę tematu, do którego został dodany wpis.

4.14 Backlog Item 14

Tytuł zadania. Utworzenie przycisku dodającego nowy wpis.

Opis zadania. Django nie tworzy przycisku wysyłającego formularz, więc definiujemy go sami.

Priorytet. Wysoki.

Definition of Done. Przycisk dodający nowy wpis.

4.15 Backlog Item 15

Tytuł zadania. Edycja wpisów.

Opis zadania. Przygotujemy stronę pozwalającą użytkownikom na edytowanie istniejących wpisów. Nakazujemy Django utworzenie formularza wypełnionego informacjami pochodzącymi z istniejącego wpisu.

Priorytet. Wysoki.

Definition of Done. Możliwość edycji istniejących wpisów przez wszystkich użytkowników.

4.16 Backlog Item 16

Tytuł zadania. Utworzenie przekierowania na stronę tematu po udanej operacji edytowania wpisu.

Opis zadania. Po edycji istniejącego wpisu użytkownik powinien zobaczyć uaktualnioną wersję zmodyfikowanego wpisu.

Priorytet. Wysoki.

Definition of Done. Przekierowanie użytkownika po edycji wpisu na stronę tematu, w którym został zmodyfikowany wpis.

4.17 Backlog Item 17

Tytuł zadania. Utworzenie przycisku zatwierdzającego zmiany w edytowanym wpisie.

Opis zadania. Django nie tworzy przycisku wysyłającego formularz, więc definiujemy go sami.

Priorytet. Wysoki.

Definition of Done. Przycisk zapisujący zmiany dokonane we wpisie.

4.18 Backlog Item 18

Tytuł zadania. Strona logowania użytkownika.

Opis zadania. Użycie oferowanych przez Django narzędzi do budowania formularzy, aby przygotować stronę pozwalającą każdemu użytkownikowi na zalogowanie się do naszej aplikacji internetowej.

Priorytet. Wysoki.

Definition of Done. Możliwość zalogowania się na swoje konto przez wszystkich użytkowników.

4.19 Backlog Item 19

Tytuł zadania. Utworzenie przekierowania na stronę główną po zakończonym sukcesem procesie logowania.

Opis zadania. Po zakończonym sukcesem logowaniu użytkownik kierowany jest z powrotem na stronę główną.

Priorytet. Średni.

Definition of Done. Przekierowanie użytkownika po udanym logowaniu na stronę główną aplikacji.

4.20 Backlog Item 20

Tytuł zadania. Utworzenie przycisku przesyłającego formularz logowania.

Opis zadania. Django nie tworzy przycisku wysyłającego formularz, więc definiujemy go sami.

Priorytet. Wysoki.

Definition of Done. Przycisk wysyłający formularz logowania.

4.21 Backlog Item 21

Tytuł zadania. Wylogowanie się z aplikacji.

Opis zadania. Nie będziemy tworzyć oddzielnej strony do obsługi wylogowania. Kiedy użytkownik kliknie łącze, zostanie przeniesiony z powrotem na stronę główną.

Priorytet. Wysoki.

Definition of Done. Możliwość wylogowania się z aplikacji przez wszystkich użytkowników.

4.22 Backlog Item 22

Tytuł zadania. Strona rejestracji użytkownika.

Opis zadania. Użycie oferowanych przez Django narzędzi do budowania formularzy, aby przygotować stronę pozwalającą każdemu użytkownikowi zarejestrować się w naszej aplikacji internetowej.

Priorytet. Wysoki.

Definition of Done. Możliwość zarejestrowania własnego konta użytkownika.

4.23 Backlog Item 23

Tytuł zadania. Utworzenie przekierowania na stronę główną po zakończonym sukcesem procesie rejestracji.

Opis zadania. Po rejestracji użytkownik kierowany jest z powrotem na stronę główną, na której spersonalizowane powitanie informuje o zakończonej sukcesem rejestracji.

Priorytet. Średni.

Definition of Done. Przekierowanie użytkownika po rejestracji na stronę główną aplikacji.

4.24 Backlog Item 24

Tytuł zadania. Utworzenie przycisku przesyłającego formularz rejestracji.

Opis zadania. Django nie tworzy przycisku wysyłającego formularz, więc definiujemy go sami.

Priorytet. Wysoki.

Definition of Done. Przycisk wysyłający formularz rejestracji.

4.25 Backlog Item 25

Tytuł zadania. Umożliwienie użytkownikom bycia właścicielami swoich danych.

Opis zadania. Ograniczymy dostęp do wybranych stron, aby użytkownik mógł pracować jedynie z własnymi danymi.

Priorytet. Wysoki.

Definition of Done. Użytkownicy mogą widzieć jedynie własne dane.

4.26 Backlog Item 26

Tytuł zadania. Zdefiniowanie interaktywnego paska nawigacji.

Opis zadania. Na pasku definiujemy zestaw łączy umożliwiających użytkownikom poruszanie się po aplikacji. Razem z paskiem nawigacji zdefiniujemy przycisk, który będzie wyświetlany, gdy okno przeglądarki będzie miało zbyt małą szerokość, aby wyświetlić poziomo cały pasek nawigacji. Kiedy przycisk ten zostanie kliknięty przez użytkownika, elementy nawigacyjne pojawią się na rozwijanej liście.

Priorytet. Wysoki.

Definition of Done. Przycisk zwijający pasek nawigacji, gdy użytkownik zmniejszy okno przeglądarki lub gdy wyświetli naszą aplikację internetową na urządzeniu mobilnym wyposażonym w mały ekran.

4.27 Backlog Item 27

Tytuł zadania. Nadanie stylu stronie głównej.

Opis zadania. Na stronie głównej aplikacji umieszczamy krótki slogan, aby użytkownik korzystający z aplikacji po raz pierwszy domyślił się od razu, jakie jest jej przeznaczenie. Zachęcamy potencjalnych użytkowników do utworzenia konta oraz przedstawiamy dwie najważniejsze akcje - dodawanie nowych tematów i wpisów. Wygląd i układ strony głównej dostosowuje się automatycznie do rozmiaru okna przeglądarki, na której jest wyświetlana aplikacja.

Priorytet. Średni.

Definition of Done. Atrakcyjność i responsywność strony głównej aplikacji.

4.28 Backlog Item 28

Tytuł zadania. Nadanie stylów stronom logowania i rejestracji.

Opis zadania. Wprowadzamy zmiany w formularzach logowania i rejestracji, aby zapewnić im spójny wygląd z pozostałymi stronami. Strony logowania i rejestracji staną się bardziej przejrzyste i czytelne (od razu wiadomo do czego służą).

Priorytet. Średni.

Definition of Done. Atrakcyjność i responsywność stron logowania i rejestracji.

4.29 Backlog Item 29

Tytuł zadania. Nadanie stylu stronie tematów.

Opis zadania. Nadajemy stronie tematów odpowiedni styl.

Priorytet. Średni.

Definition of Done. Atrakcyjność i responsywność strony tematów.

4.30 Backlog Item 30

Tytuł zadania. Nadanie stylów wpisom na stronie tematu.

Opis zadania. Nadajemy wpisom na stronie tematu odpowiedni styl.

Priorytet. Średni.

Definition of Done. Atrakcyjność i responsywność wpisów na stronie tematu.

4.31 Backlog Item 31

Tytuł zadania. Wdrożenie aplikacji "Memorize!".

Opis zadania. Wdrażamy projekt na serwerze WWW, aby każdy, kto ma połączenie z internetem mógł skorzystać z naszej aplikacji.

Priorytet. Wysoki.

Definition of Done. Działająca online aplikacja "Memorize!".

5 Sprint 1

5.1 Cel

Do opracowania projektu o nazwie "Memorize!" wykorzystamy framework Django. Zaczniemy od przygotowania środowiska dla projektu, a następnie zdefiniujemy klasy dla danych, z którymi będzie działała aplikacja. Witrynę administracyjną Django wykorzystamy do wstawienia pewnych danych początkowych, a następnie przejdziemy do tworzenia widoków i szablonów, za pomocą których Django będzie budować strukturę naszej aplikacji internetowej. Utworzymy intuicyjne, przyjazne użytkownikom strony pozwalające im na dodawanie nowych tematów oraz wpisów, a także na edycję istniejących wpisów.

5.2 Sprint Planning/Backlog

Tytuł zadania. Instalacja i konfiguracja Django.

- Estymata: "L".

Tytuł zadania. Utworzenie bazy danych.

- Estymata: "S".

Tytuł zadania. Utworzenie i konfiguracja konta super - użytkownika.

- Estymata: "M".

Tytuł zadania. Zdefiniowanie odpowiednich klas.

- Estymata: "XL".

Tytuł zadania. Ograniczenie długości wyświetlanych wpisów na stronie tematu.

- Estymata: "L".

Tytuł zadania. Utworzenie strony głównej naszej aplikacji internetowej.

- Estymata: "L".

Tytuł zadania. Utworzenie stron wyświetlających tematy i zawarte w nich wpisy.

- Estymata: "L".

Tytuł zadania. Utworzenie łącza ze strony tematów.

- Estymata: "M".

Tytuł zadania. Dodawanie nowych tematów.

- Estymata: "XL".

Tytuł zadania. Utworzenie przekierowania do listy tematów po dodaniu nowego tematu.

- Estymata: "M".

Tytuł zadania. Utworzenie przycisku dodającego nowy temat.

- Estymata: "S".

Tytuł zadania. Dodawanie nowych wpisów.

- Estymata: "XL".

Tytuł zadania. Utworzenie przekierowania na stronę tematu po dodaniu nowego wpisu.

- Estymata: "M".

Tytuł zadania. Utworzenie przycisku dodającego nowy wpis.

- Estymata: "S".

Tytuł zadania. Edycja wpisów.

- Estymata: "XL".

Tytuł zadania. Utworzenie przekierowania na stronę tematu po udanej operacji edytowania wpisu.

- Estymata: "M".

Tytuł zadania. Utworzenie przycisku zatwierdzającego zmiany w edytowanym wpisie.

- Estymata: "S".

5.3 Realizacja

Tytuł zadania. Instalacja i konfiguracja Django.

Wykonawca. Zespół deweloperski.

Realizacja. Wykonujemy polecenie instalacyjne frameworka Django na swoich systemach operacyjnych. Kolejnymi poleceniami nakazujemy Django utworzyć nowy projekt o nazwie memorize, a następnie strukturę katalogów ułatwiającą wdrożenie aplikacji na serwerze, gdy zakończymy już nad nią pracę. Czasochłonność tego zadania była przesadzona. Ostateczna estymata mierzona w koszulkach wyniosła "M".

Tytuł zadania. Utworzenie bazy danych.

Wykonawca. Zespół deweloperski.

Realizacja. Wydajemy po raz pierwszy polecenie nakazujące Django sprawdzić, czy baza danych odpowiada bieżącemu stanowi projektu. W wyniku tego polecenia Django utworzy nową bazę danych zawierającą tabele niezbędne do przechowywania informacji, których będziemy używać w projekcie. Upewniamy się, że Django utworzył plik o nazwie db.sqlite3. Faktyczny czas spędzony nad ukończeniem tego zadania pokrywa się z przyjętą wcześniej estymatą "S".

Tytuł zadania. Zdefiniowanie odpowiednich klas.

Wykonawca. Damian Hadała, Krzysztof Radoń.

Realizacja. Utworzyliśmy klasę o nazwie Topic. W tej klasie mamy dwa atrybuty: text i date_added. Atrybut text jest typu CharField, co oznacza dane składające się ze znaków, czyli po prostu tekst. Podczas definiowania atrybutu typu CharField konieczne jest poinformowanie Django, jaka ilość miejsca na te informacje powinna być zarezerwowana w bazie danych. Podajemy wielkość 200 znaków (max_length=200), która powinna być wystarczająca dla praktycznie wszystkich tytułów. Atrybut date_added jest typu DateTimeField, czyli przechowuje informacje o dacie i godzinie. Przekazujemy argument auto_add_now=True, który nakazuje Django automatyczne przypisywanie temu atrybutowi bieżącej daty i godziny w chwili tworzenia nowego tematu przez użytkownika. Utworzyliśmy klasę o nazwie Entry. Pierwszy atrybut (topic) to egzemplarz ForeignKey, czyli klucz zewnętrzny - łączy każdy wpis z określonym tematem. Następny atrybut nosi nazwę text i jest egzemplarzem TextField. Nie narzucamy mu żadnych ograniczeń dotyczących wielkości, ponieważ nie chcemy ograniczać długości poszczególnych wpisów. Atrybut date_added pozwala wyświetlać wpisy w kolejności ich tworzenia oraz umieszczać znacznik czasu obok każdego wpisu. Czas, jaki poświęciliśmy na realizację tego zadania śmiało można zamknąć we wcześniej ustalonym rozmiarze "XL".

Tytuł zadania. Utworzenie i konfiguracja konta super - użytkownika.

Wykonawca. Dawid Sułowicz.

Realizacja. Wydałem Django polecenie utworzenia konta super - użytkownika: createsuperuser i nadałem mu nazwę mm_admin. Ustawiłem też przygotowane wcześniej z właścicielem produktu hasło do konta super - użytkownika. Modyfikuję utworzony przez framework Django plik o nazwie admin.py rejestrując klasę Topic. Dzięki temu uzyskujemy możliwość dodawania nowych tematów przez witrynę administracyjną.

Tytuł zadania. Ograniczenie długości wyświetlanych wpisów na stronie tematu.

Wykonawca. Krzysztof Jamro.

Realizacja. Na końcu klasy Entry umieszczam metodę __str__(), która wskazuje jakie informacje mają zostać wyświetlone podczas odwoływania się do poszczególnych wpisów. Cały wpis nie może być w postaci długiego tekstu więc wyświetlam jedynie pierwszych 50 znaków przechowywanych w atrybucie text. Dołączam również wielokropek, aby tym wyraźnie wskazać, że nie został wyświetlony pełny wpis.

Tytuł zadania. Utworzenie strony głównej naszej aplikacji internetowej.

Wykonawca. Damian Hadała.

Realizacja. Moim zadaniem jest zdefiniowanie adresu URL dla strony głównej, utworzenie jej funkcji widoku oraz stworzenie prostego szablonu. Obecnie bazowy adres URL prowadzi strony domyślnej Django informującej o prawidłowym działaniu projektu. Zmieniam to mapując ten bazowy adres URL na stronę główną aplikacji "Memorize!":

```
"""Wzorce adresów URL dla memorizeapp"""

from django.conf.urls import url

from . import views

urlpatterns = [
    #Strona główna
    url(r'^$', views.index, name='index'),
]
```

Następnie tworzę widok dla strony głównej naszej aplikacji:

```
def index(request):
    """Strona główna dla aplikacji Memorize!"""
return render(request, 'memorizeapp/index.html')
```

Kiedy adres URL żądania zostanie dopasowany do wzorca, Django będzie szukać w pliku views.py funkcji o nazwie index(). Funkcji tej zostanie przekazany obiekt request. Funkcja render(), którą tutaj użyłem ma dwa argumenty: pierwotny obiekt request i szablon, który będzie wykorzystywany do wyświetlania strony głównej. Przystępuję teraz do utworzenia szablonu. Tworzę plik index.html, a następnie zamieszczam w nim dwa akapity: pierwszy jest rodzajem tytułu, natomiast drugi wyjaśnia użytkownikowi przeznaczenie naszej aplikacji. Teraz po wykonaniu żądania do bazowego adresu URL projektu, zamiast strony domyślnie generowanej przez Django wyświetli się szablon, który właśnie zbudowałem.

Tytuł zadania. Utworzenie stron wyświetlających tematy i zawarte w nich wpisy.

Wykonawca. Krzysztof Jamro, Krzysztof Radoń, Damian Hadała.

Realizacja. Rozpoczynamy od utworzenia szablonu bazowego o nazwie base.html, który będzie zawierał elementy stosowane na wszystkich stronach, a pozostałe szablony będą dziedziczyły po tym szablonie. Na początek umieszczamy w nim w nim tytuł i łącze prowadzące do strony głównej aplikacji. Od teraz wszystkie strony naszej aplikacji będą miały łącze prowadzące na stronę główną. Następnie modyfikujemy plik szablonu index.html w taki sposób, aby dziedziczył po base.html. Wszystko to, co nie jest dziedziczone po szablonie nadrzędnym zostaje umieszczone wewnątrz bloku content. My umieszczamy tam akapit opisujący aplikację "Memorize!". Przygotujemy teraz następujące strony: pierwsza z nich to ogólna strona tematów, natomiast druga posłuży do wyświetlania wpisów dla pojedynczego tematu. Definiujemy najpierw adres URL dla strony tematów. Powszechną praktyką jest wybieranie takiego adresu URL, żeby odzwierciedlał rodzaj informacji przedstawianych na danej stronie.

My wykorzystaliśmy słowo topics. Po wpisaniu w przeglądarce adresu w postaci /topics/ przejdziemy na stronę z tematami. Teraz przygotowujemy zapytanie do bazy danych: interesują nas obiekty Topic posortowane według daty dodania. Przechowujemy ten zbiór w zmiennej topics. Tworzymy teraz szablon tematów o nazwie topics.html i w nim wykorzystamy dane dostarczone przez funkcję widoku topics(). Dziedziczymy po szablonie bazowym i rozpoczynamy wypełnianie bloku content. Na tej stronie znajdzie się wypunktowana lista wszystkich tematów. Pokazujemy też Django co ma zrobić, jeśli lista tematów nie będzie zawierała żadnych elementów. Zostanie wyświetlony komunikat o tym, że użytkownik nie dodał jeszcze żadnego tematu. Modyfikujemy teraz szablon bazowy w taki sposób, aby zawierał łącze do strony wyświetlającej tematy. Następnie naszym zadaniem jest utworzenie strony, która ma wyświetlić nazwę tematu oraz wszystkie powiązane z nim wpisy. We wzorcu adresu URL wykorzystujemy atrybut id w celu wskazania żądanego tematu. Tworzymy funkcję widoku topic(), która pobierze z bazy danych temat oraz wszystkie powiązane z nim wpisy i wyświetli je w kolejności od najnowszego. Przedostatnim zadaniem, które realizujemy jest utworzenie kolejnego szablonu, tym razem dla strony tematu, który powinien wyświetlać nazwę tematu oraz powiązane z nim wpisy. Wyświetlamy wszystkie wpisy za pomocą wypunktowanej listy. Każdy dodany wpis zawiera znacznik czasu w formacie 1 sty 2017 19:00. Jeśli do danego tematu nie został dodany żaden wpis, to użytkownik zostanie o tym poinformowany. Ostateczna estymata nie wyniosła "L", tak jak to było zaplanowane, tylko "XL" co znacznie wydłużyło pracę nad zadaniem.

Tytuł zadania. Utworzenie łącza ze strony tematów.

Wykonawca. Krzysztof Jamro.

Realizacja. Aby móc wyświetlić stronę danego tematu w przeglądarce modyfikuję szablon tematów topics.html tak, aby każde łącze tematu prowadziło na odpowiednią stronę. Od teraz każdy temat na liście jest łączem, którego kliknięcie powoduje przejście na odpowiednią stronę. Zadanie zostało wykonane wcześniej, niż przypuszczałem. Końcowa estymata tego zadania wyniosła "S".

Tytuł zadania. Dodawanie nowych tematów.

Wykonawca. Zespół deweloperski.

Realizacja. Tworzymy formularz dodawania nowego tematu i umieszczamy go w pliku forms.py, który znajduje się w katalogu memorizeapp. Definiujemy klasę o nazwie TopicForm. Budujemy ten formularz na podstawie klasy Topic i uwzględniamy jedynie kolumny text. Ustalamy adres URL dla strony new_topic. Powinien być krótki i jasny. Kiedy użytkownik będzie chciał dodać nowy temat zostanie przekierowany na stronę/new_topic/. Następnie tworzymy funkcję widoku new_topic(), która będzie się zajmować przetworzeniem danych wpisanych przez użytkownika w formularzu. Na koniec tworzymy szablon o nazwie new_topic.html przeznaczony do wyświetlenia zbudowanego przez nas formularza oraz łącze prowadzące na stronę new_topic. To zadanie nie okazało się aż tak czasochłonne. Końcowa estymata wyniosła "L".

Tytuł zadania. Utworzenie przekierowania do listy tematów po dodaniu nowego tematu.

Wykonawca. Dawid Sułowicz.

Realizacja. Importuję klasę HttppResponseRedirect i przy jej pomocy ustalam przekierowanie użytkownika, kiedy już doda nowy temat, z powrotem na stronę topics:

```
from django.http import HttpResponseRedirect
--cięcie--
return HttpResponseRedirect(reverse('memorizeapp:topics'))
```

Funkcja reverse() określa adres URL na podstawie wzorca adresu, co znaczy, że Django wygeneruje adres URL podczas żądania strony. Jako, że spędziłem trochę czasu w dokumentacji Django, aby ukończyć to zadanie, to pierwsze przekierowanie mogę ocenić na zgodne z estymatą "M". Pozostałym członkom zespołu jednak będę już proponował ocenę czasochłonności "S".

Tytuł zadania. Utworzenie przycisku dodającego nowy temat.

Wykonawca. Dawid Sułowicz.

Realizacja. Definiuję przycisk dla szablonu strony o nazwie new_topic.html, ponieważ Django nie tworzy automatycznie przycisku wysyłającego formularz:

```
<button name="submit">Dodaj temat</button>
```

Z tym zadaniem nie miałem najmniejszego problemu. Czasochłonność, jaką to zadanie generowało jest zgodna z wcześniejszymi założeniami, czyli "S".

Tytuł zadania. Dodawanie nowych wpisów.

Wykonawca. Zespół deweloperski.

Realizacja. Tworzymy formularz dodawania nowych wpisów, który będzie powiązany z klasą Entry. Zaczynamy od uaktualnienia polecenia import, aby importowało nie tylko klasę Topic, ale również Entry. Utworzona klasa EntryForm określa kolumny przeznaczone do uwzględnienia w formularzu. W adresie URL przeznaczonym do dodawania nowego wpisu umieszczamy argument topic_id, aby każdy wpis był powiązany z określonym tematem. Tworzymy nową funkcję widoku o nazwie new_entry() odpowiedzialną za obsługę dodawania nowych wpisów. Dzięki niej wpis zostanie zapisany w bazie danych wraz z przypisanym mu tematem. Tworzymy szablon dla strony new_entry, który jest bardzo podobny do utworzonego przez nas wcześniej szablonu new_topic. Temat jest wyświetlany na początku strony, aby użytkownik nie miał żadnych wątpliwości, do jakiego tematu dodaje nowy wpis. Na samym końcu dodajemy łącze prowadzące na stronę new_entry, które będzie wyświetlane na wszystkich stronach tematów. To zadanie wcześniej oszacowaliśmy na czasochłonność równą rozmiarowi "XL", co jednak okazało się przeszacowaniem. Rzeczywisty czas realizacji tego zadania zamyka się w estymacie równej "L".

Tytuł zadania. Utworzenie przekierowania na stronę tematu po dodaniu nowego wpisu.

Wykonawca. Krzysztof Radoń.

Realizacja. Przygotowuję przekierowanie użytkownika na stronę tematu, do którego został dodany nowy wpis. Na tej stronie użytkownik powinien zobaczyć nowy wpis na liście wszystkich wpisów dla danego tematu:

```
return HttpResponseRedirect(reverse('memorizeapp:topic',
args=[topic_id]))
```

Wywołanie reverse() wymaga dwóch argumentów. Pierwszy to nazwa wzorca adresu URL, na podstawie którego zostaje gnenerowany adres URL. Drugi to lista args zawierająca wszystkie argumenty, które muszą być umieszczone w adresie URL. Estymata tego zadania okazała się być mniejsza, niż zakładaliśmy. Ostatecznie jest to rozmiar "S".

Tytuł zadania. Utworzenie przycisku dodającego nowy wpis.

Wykonawca. Dawid Sułowicz.

Realizacja. Tak, jak w przypadku szablonu new_topic.html tworzę przycisk wysyłający formularz dla strony dodawania nowych wpisów new_entry.html:

```
<button name='submit'>Dodaj wpis</button>
```

Podobnie, jak poprzednim razem to zadanie nie wymagało poświęcenia dużej ilości czasu. Czasochłonność zadania oceniam na "S" i jest to zgodne z wcześniejszymi przypuszczeniami zespołu.

Tytuł zadania. Edycja wpisów.

Wykonawca. Zespół deweloperski.

Realizacja. Przygotujemy stronę, która będzie pozwalać użytkownikowi na edytowanie istniejącego wpisu. Najpierw przygotowujemy adres URL dla strony edycji wpisu. Wymaga on przekazania identyfikatora wpisu, który chcemy edytować. Następnie tworzymy funkcję widoku edit_entry(), której wartością zwrotną będzie formularz przeznaczony do edycji wpisu. Po otrzymaniu żądania, które będzie zawierać zmodyfikowany tekst wpisu, funkcja edit_entry() zapisze ten tekst w bazie danych. Po utworzeniu funkcji widoku przechodzimy do tworzenia szablonu dla strony edycji wpisu. Jest on bardzo podobny do szablonu new_entry.html. Ostatnim krokiem jest dodanie łącza prowadzącego na stronę edycji wpisu i wyświetlanego przy wszystkich wpisach na stronie tematu. Łącze pozwalające na edycje wpisu umieściliśmy po dacie i tekście danego wpisu. Zgodnie z wcześniejszymi szacowaniami to zadanie pochłonęło słuszną ilość czasu równą estymacie o rozmiarze "XL".

Tytuł zadania. Utworzenie przekierowania na stronę tematu po udanej operacji edytowania wpisu.

Wykonawca. Krzysztof Radoń.

Realizacja. W funkcji widoku edit_entry() zamieszczam przekierowanie na stronę tematu przy pomocy zaimportowanej klasy HttpResponseRedirect. Po przekierowaniu użytkownik powinien zobaczyć uaktualnioną wersję edytowanego wpisu:

```
return HttpResponseRedirect(reverse('memorizeapp:topic', args=[topic.id]))
```

Czasochłonność tego zadania okazała się być mniejsza, niż szacowaliśmy. Realny czas, który poświęciłem na jego realizację zamyka się w rozmiarze "S".

Tytuł zadania. Utworzenie przycisku zatwierdzającego zmiany w edytowanym wpisie.

Wykonawca. Dawid Sułowicz.

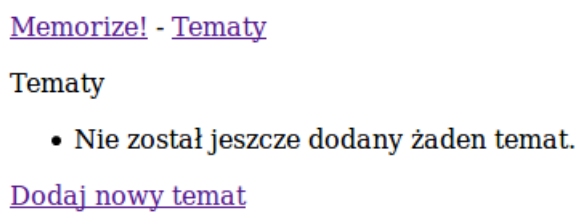
Realizacja. Tworzę przycisk wysyłający formularz, który będzie miał etykietę Zapisz zmiany, aby przypomnieć użytkownikowi, że zapisuje zmiany dokonane w istniejącym wpisie, a nie tworzy nowy wpis:

```
<button name="submit">Zapisz zmiany</button>
```

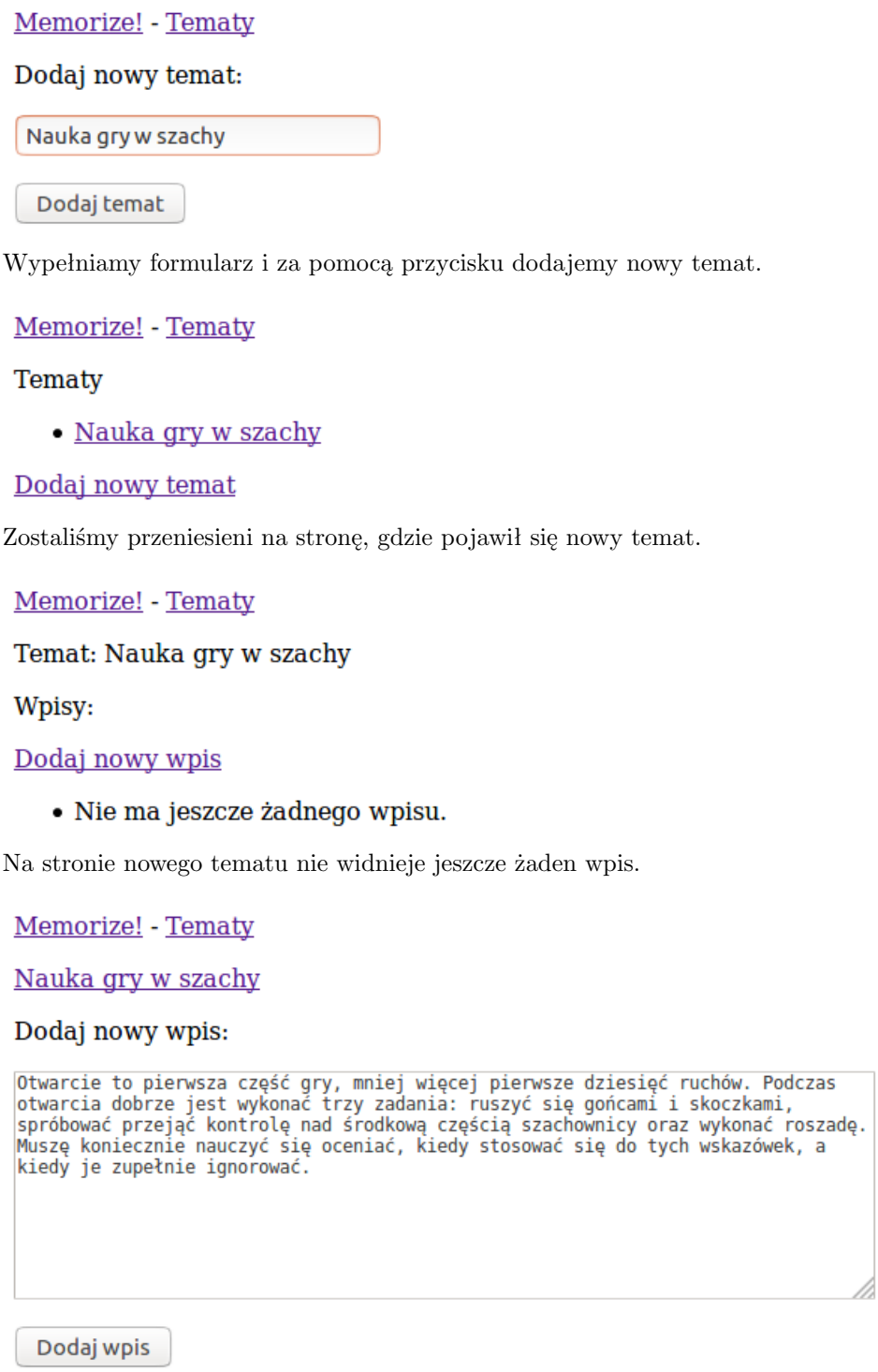
Rzeczywista czasochłonność tego zadania okazała się potwierdzać nasze przypuszczenia. Jej rozmiar mierzony w koszulkach to "S".

5.4 Sprint Review/Demo

Pierwszy sprint został zakończony powodzeniem. Wszystkie założone zadania zostały zrealizowane, choć estymaty niektórych z nich były niedokładnie oszacowane. Zrealizowaliśmy wszystkie zaplanowane na ten sprint Backlog Item'y. Utworzyliśmy surowe, ale funkcjonalne strony pozwalające użytkownikom dodawać nowe tematy oraz wpisy, a także edytować istniejące wpisy. Zakładany przyrost projektu został osiągnięty. Demonstracja uzyskanego przyrostu:



Widzimy, że na liście nie figuruje jeszcze żaden temat.

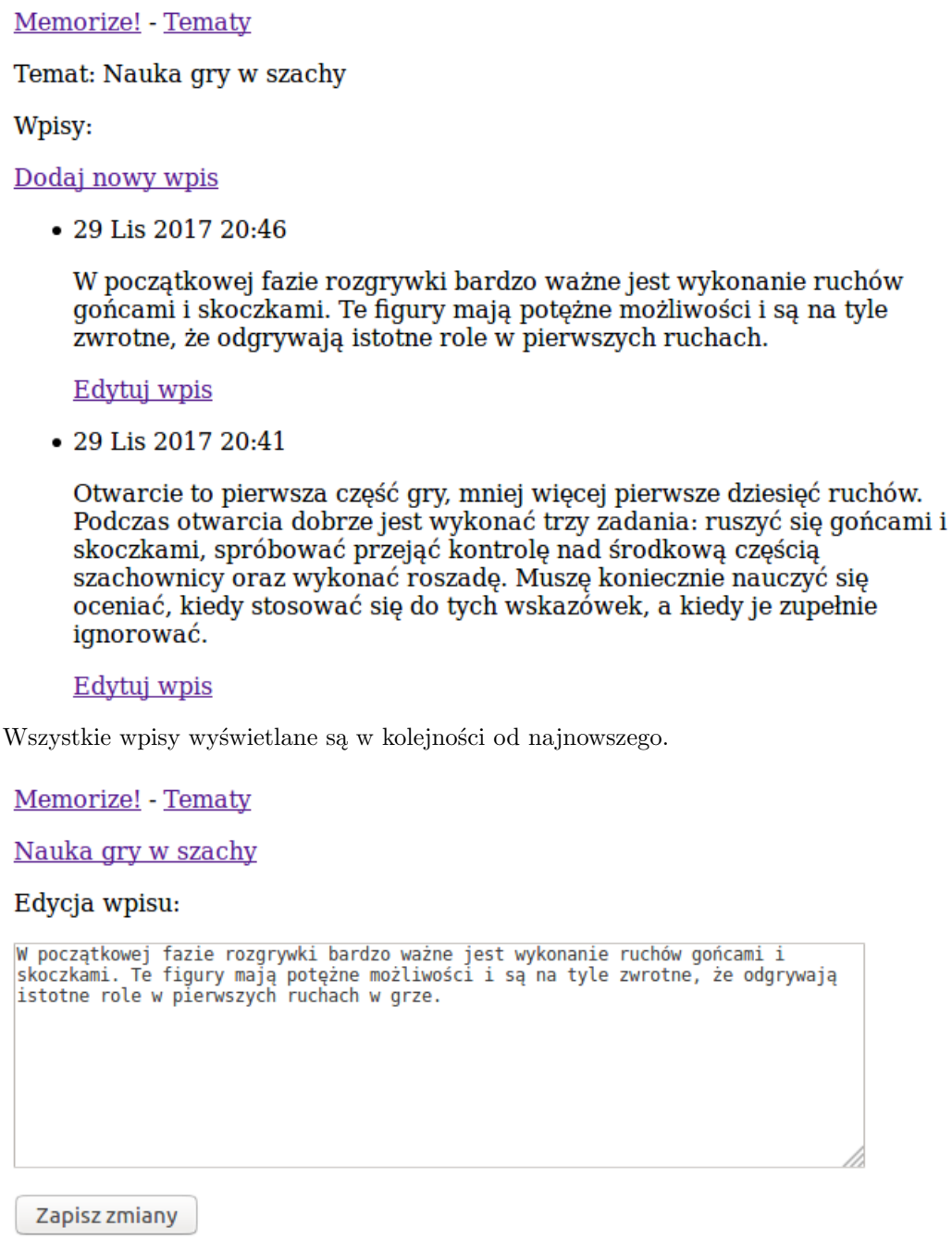


Wypełniamy formularz i za pomocą przycisku dodajemy nowy temat.

Zostaliśmy przeniesieni na stronę, gdzie pojawił się nowy temat.

Na stronie nowego tematu nie widnieje jeszcze żaden wpis.

Dodajemy nowy wpis i zamieszczamy go na stronie tematu.



Wszystkie wpisy wyświetlane są w kolejności od najnowszego.

W dowolnej chwili możemy edytować zamieszczony wcześniej wpis.

6 Sprint 2

6.1 Cel

Zajmiemy się budowaniem systemu rejestracji kont użytkowników oraz ich autoryzacji, który pozwoli użytkownikom na tworzenie kont i wykorzystywanie na-

szej aplikacji internetowej. Umożliwimy każdemu użytkownikowi znajdującemu się w dowolnym miejscu na świecie przeprowadzenie operacji rejestracji konta w aplikacji "Memorize!" i rozpoczęcie korzystania z niej. Następnie przystąpimy do implementacji systemu uwierzytelniania użytkownika. Zbudujemy stronę rejestracji, która umożliwi użytkownikom założenie konta, oraz ograniczy dostęp do wybranych stron, które będą mogli wyświetlić tylko zalogowani użytkownicy. Zmodyfikujemy niektóre funkcje widoku, aby użytkownicy mogli widzieć jedynie własne dane.

6.2 Sprint Planning/Backlog

Tytuł zadania. Strona logowania użytkownika.

- Estymata: "XL".

Tytuł zadania. Utworzenie przekierowania na stronę główną po zakończonym sukcesem procesie logowania.

- Estymata: "M".

Tytuł zadania. Utworzenie przycisku przesyłającego formularz logowania.

- Estymata: "S".

Tytuł zadania. Wylogowanie się z aplikacji.

- Estymata: "M".

Tytuł zadania. Strona rejestracji użytkownika.

- Estymata: "XL".

Tytuł zadania. Utworzenie przekierowania na stronę główną po zakończonym sukcesem procesie rejestracji.

- Estymata: "M".

Tytuł zadania. Utworzenie przycisku przesyłającego formularz rejestracji.

- Estymata: "S".

Tytuł zadania. Umożliwienie użytkownikom bycia właścicielami swoich danych.

- Estymata: "XL".

6.3 Realizacja

Tytuł zadania. Strona logowania użytkownika.

Wykonawca. Krzysztof Radoń.

Realizacja. Podczas implementacji strony logowania wykorzystałem domyślny widok *login* wbudowany w Django. Tworzę nowy plik *urls.py* w folderze *memorize/users* i dodaję nieco różniący się od poprzednich wzorzec adresu URL dla strony logowania:

```
url(r'^login/$', login, {'template_name': 'users/login.html'},
name='login')
```

Teraz kiedy użytkownika zażąda wyświetlenia strony logowania, domyślnie Django wykorzysta wbudowany widok *login*. Nadal jednak trzeba dostarczać informację o szablonie dla tej strony. W podkatalogu *templates* umieszczam plik szablonu dla strony logowania. Ten szablon rozszerza szablon bazowy *base.html*, co gwarantuje, że strona logowania będzie miała taki sam wygląd i sposób działania jak pozostała część aplikacji "Memorize!". W poniższym fragmencie szablonu strony logowania sprawdzam, czy wystąpiły błędy podczas wypełniania formularza logowania:

```
{% if form.errors %}
  <p>Nazwa użytkownika i hasło są nieprawidłowe.
  Proszę spróbować ponownie.</p>
{% endif %}
```

Jeżeli wystąpi błąd, formularz wyświetli komunikat błędu informujący, że podana nazwa użytkownika i hasło nie pasują do informacji przechowywanych w bazie danych. Następnie w szablonie bazowym *base.html* umieszczam łącze prowadzące na stronę logowania, aby było wyświetlane na każdej stronie.

Tytuł zadania. Utworzenie przekierowania na stronę główną po zakończonym sukcesem procesie logowania.

Wykonawca. Krzysztof Radoń.

Realizacja. Korzystam z wbudowanej funkcji widoku login i ustanawiam przekierowanie na stronę główną. Teraz użytkownik po udanym zalogowaniu zostanie przeniesiony na stronę główną aplikacji.

Tytuł zadania. Utworzenie przycisku przesyłającego formularz logowania.

Wykonawca. Krzysztof Radoń.

Realizacja. Kiedy formularz zostanie wyświetlony potrzebny nam się przycisk odpowiedzialny za przesłanie wypełnionego formularza. W szablonie dla strony logowania *login.html* umieszczam przycisk wysyłający ten formularz:

```
<button name="submit">Zaloguj się</button>
```

Tytuł zadania. Wylogowanie się z aplikacji.

Wykonawca. Krzysztof Jamro.

Realizacja. Na samym początku definiuję wzorzec adresu URL dla łącza wylogowania. Ten wzorzec adresu zostanie przekazany do funkcji wylogowania. Tworzę funkcję widoku logout_view() i importuję wbudowaną w Django funkcję logout(). Za jej pomocą użytkownik będzie miał możliwość wylogowania się i zaraz po wylogowaniu zostanie przeniesiony na stronę główną. Na koniec w bazowym szablonie dodaję łącze do wylogowania, które będzie widoczne na każdej stronie:

```
<a href="{% url 'users:logout' %}">Wyloguj się</a>
```

Tytuł zadania. Strona rejestracji użytkownika.

Wykonawca. Damian Hadała, Krzysztof Radoń, Dawid Sułowicz.

Realizacja. Do utworzenia strony rejestracji użytkownika posłuży nam wbudowany w Django formularz *UserCreationForm*. Sam formularz nie wystarczy. Utworzymy własną funkcję widoku i szablon. Najpierw jednak dodamy wzorzec URL dla strony rejestracji w pliku *urls.py*. Ten wzorzec przekażę żądania do funkcji rejestracji, którą teraz utworzymy. Nasza funkcja rejestracji będzie się nazywać *register()*. Funkcja będzie wyświetlać pusty formularz rejestracji, a później przetwarzać go, jak zostanie już wypełniony. Dzięki tej funkcji będzie odbywać się rejestracja użytkownika.

Tytuł zadania. Utworzenie przekierowania na stronę główną po zakończonym sukcesem procesie rejestracji.

Wykonawca. Dawid Sułowicz.

Realizacja. Już po udanej operacji rejestracji nowego użytkownika, przekierowujemy go na stronę główną. Na niej użytkownik zobaczy spersonalizowane powitanie informującego go zakończonej sukcesem rejestracji.

Tytuł zadania. Utworzenie przycisku przesyłającego formularz rejestracji.

Wykonawca. Krzysztof Jamro.

Realizacja. W szablonie przeznaczonym dla strony rejestracji użytkownika o nazwie register.html zamieszczam przycisk, za pomocą którego zostanie przesłany formularz rejestracji:

```
<button name="submit">Zarejestruj się</button>
```

Za pomocą metody as_p upewniam się, że framework będzie prawidłowo wyświetlał wszelkie komunikaty błędów, które mogą się zdarzyć, kiedy formularz nie będzie poprawnie wypełniony.

Tytuł zadania. Umożliwienie użytkownikom bycia właścicielami swoich danych.

Wykonawca. Krzysztof Radoń.

Realizacja. Za pomocą dekoratora *@login_required* ograniczam dostęp do strony tematów. Teraz tylko zarejestrowani użytkownicy mają dostęp do listy tematów. Kod funkcji *@login_required* sprawdza, czy użytkownik jest zalogowany, ponieważ jest to warunek konieczny do wykonania funkcji topics():

```
@login_required
def topics(request):
"""Wyświetlenie wszystkich tematów"""
--cięcie--
```

Jeżeli użytkownik nie jest zalogowany, to zostanie przekierowany na stronę logowania. W naszej aplikacji pozostawimy jako publicznie dostępne strony: główną, rejestracji i wylogowania. Dla wszystkich pozostałych stron używam dekoratora *@login_required*. Wprowadzam teraz zmianę w funkcji widoku tematów polegającą na tym, że użytkownik będzie mógł wyświetlać jedynie tematy utworzone przez siebie. Aby uniknąć możliwości podglądu tematów przez użytkowników mających adres tematu innego użytkownika, przeprowadzam operację weryfikacji jeszcze przed pobraniem żądanych wpisów w funkcji widoku tematu:

```
#Sprawdzenie poziomu uprawnień
if topic.owner != request.user:
raise Http404
```

Jeżeli bieżący użytkownik nie będzie właścicielem żądanego tematu, Django zwróci stronę z kodem błędu 404. Odpowiedź 404 to standardowa odpowiedź błędu zwracana w sytuacji, gdy żądany zasób nie istnieje na serwerze. Tak samo postępuję ze stronami edycji i dodawania nowego wpisu. Na koniec modyfikuję kod funkcji *new_topic()* tak, aby każdy nowo tworzony temat został powiązany z bieżącym użytkownikiem.

6.4 Sprint Review/Demo

Drugi sprint został zakończony powodzeniem. Zrealizowaliśmy wszystkie zaplanowane na ten sprint Backlog Item'y. Możemy teraz dodawać dowolną liczbę nowych tematów dla różnych użytkowników. Każdy użytkownik zachowa dostęp jedynie do własnych danych niezależnie od rodzaju przeprowadzanych na nich operacji: przeglądanie wpisów, tworzenie nowych lub modyfikowanie tych już istniejących.

7 Sprint 3

7.1 Cel

Nadamy styl aplikacji "Memorize!", aby prezentowała sie profesjonalnie na wszystkich nowoczesnych urządzeniach, począwszy od tych podłączonych do ogromych płaskich monitorów aż po smartfony.

7.2 Sprint Planning/Backlog

Tytuł zadania. Zdefiniowanie interaktywnego paska nawigacji.

- Estymata: "XL".

Tytuł zadania. Nadanie stylu stronie głównej.

- Estymata: "L".

Tytuł zadania. Nadanie stylów stronom logowania i rejestracji.

- Estymata: "L".

Tytuł zadania. Nadanie stylu stronie tematów.

- Estymata: "M".

Tytuł zadania. Nadanie stylów wpisom na stronie tematu.

- Estymata: "L".

7.3 Realizacja

Tytuł zadania. Zdefiniowanie interaktywnego paska nawigacji.

Wykonawca. Krzysztof Radoń.

Realizacja. Definiuję element *nav*, który wskazuje na początek sekcji zawierającej łącza nawigacyjne, które mają znaleźć się na stronie. Następnie definiuję przycisk, który będzie wyświetlany, gdy okno przeglądarki będzie miało zbyt małą szerokość, aby wyświetlić poziomo cały pasek nawigacji. Kiedy użytkownik aplikacji naciśnie ten przycisk, elementy nawigacyjne pojawią się na rozwijanej liście. Po lewej stronie paska nawigacyjnego umieszczam nazwę aplikacji, która jest jednocześnie łączem prowadzącym do strony głównej. Przygotowuję zestaw łączy umożliwiających użytkownikom poruszanie się po aplikacji internetowej. Pasek nawigacji to w zasadzie lista zaczynająca się od znacznika *ul*. Przy prawej krawędzi paska nawigacyjnego zamieszczam łącza rejestracji i logowania. Estymacja czasu trwania tego zadania jest zgodna z wcześniejszymi założeniami.

Tytuł zadania. Nadanie stylu stronie głównej.

Wykonawca. Damian Hadała.

Realizacja. Do uaktualnienia strony głównej używam Bootstrapa i jego elementu o nazwie jumbotron, czyli dużego prostokąta, który będzie się wyróżniał na stronie. Wewnątrz tego nowego elementu umieszczam chwytliwy slogan, aby użytkownik odwiedzający naszą witrynę w internecie po raz pierwszy domyślił się od razu, jakie jest przeznaczenie aplikacji Memorize. Poniżej dodaję tekst dostarczający nieco więcej informacji. Zachęca on użytkowników do utworzenia konta oraz przedstawia dwie najważniejsze akcje, czyli dodawanie nowych tematów i wpisów. Strona główna będzie wyglądała zdecydowanie lepiej niż w pierwotnej fazie powstawania aplikacji, kiedy była pozbawiona stylów. Szacowana estymata czasowa "L" dla tego zadania okazała się być odpowiednia.

Tytuł zadania. Nadanie stylów stronom logowania i rejestracji.

Wykonawca. Krzysztof Jamro, Krzysztof Radoń.

Realizacja. Wprowadzamy zmiany w formularzach logowania i rejestracji, aby zapewnić im spójny wygląd z pozostałymi stronami. Wczytujemy do szablonów *login.html* i *register.html* znaczniki *bootstrap3*. Definiujemy blok *header*, który będzie opisywał przeznaczenie tych stron. Używamy szablonu znacznika *bootstrap_form* do wyświetlenia formularza. Za pomocą znacznika *buttons* dodajemy styl dla przycisków. Estymata określona w koszulkach jako "L" daje dobry obraz pracy włożonej w realizację tego zadania.

Tytuł zadania. Nadanie stylu stronie tematów.

Wykonawca. Dawid Sułowicz, Krzysztof Radoń.

Realizacja. Wewnątrz bloku *header* w szablonie *topics.html* dodajemy nagłówek *Tematy*. Każdy temat umieszczamy w elemencie *h3*, aby był nieco większy. Takie samo podejście stosujemy dla łącza, za pomocą którego będzie dodawany nowy temat. Określona wcześniej estymata dobrze odzwierciedla czas spędzony nad realizacją tego zadania.

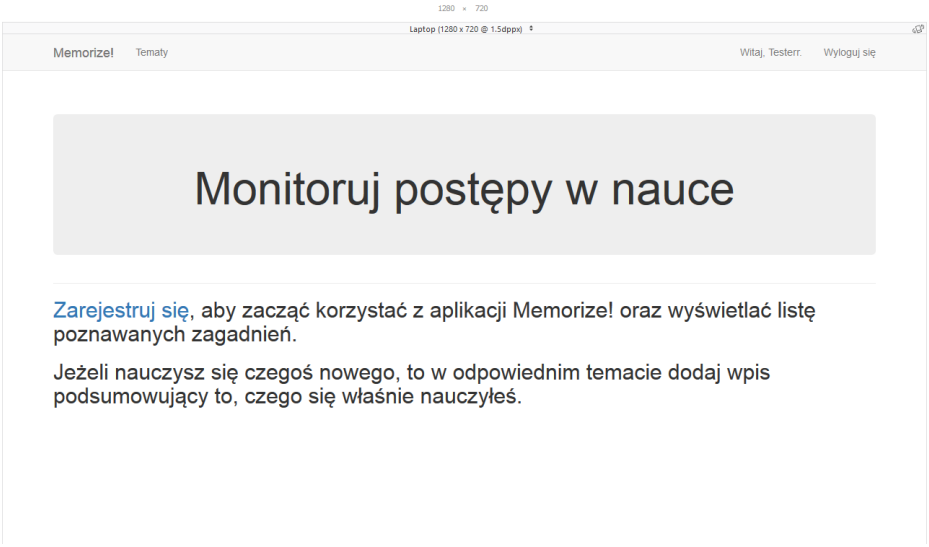
Tytuł zadania. Nadanie stylów wpisom na stronie tematu.

Wykonawca. Damian Hadała.

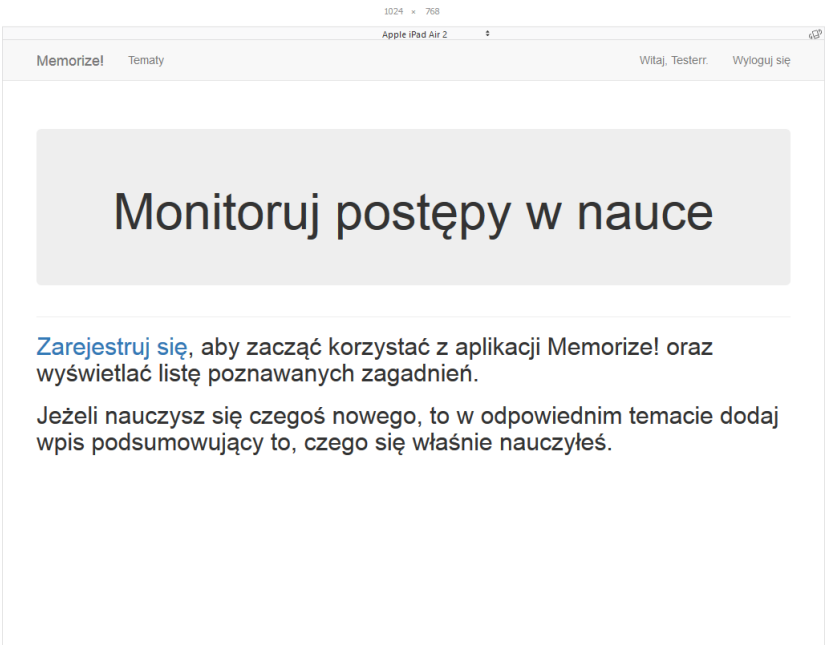
Realizacja. Do strony tematu używam paneli Bootstrapa, aby każdy wpis na stronie się wyróżniał. Ten panel dla każdego wpisu to po prostu znacznik div z predefiniowanymi stylami. Przede wszystkim temat umieszczam w nagłówku. Następnie pozbywam się struktury nieuporządkowanej listy, jaką poprzednio utworzyliśmy w szablonie tematu. Tworzę znacznik div, który zawiera dwa następne zagnieżdżone elementy. Pierwszy element zawiera datę wpisu oraz łącze pozwalające na jego edycję. Aby wyświetlić te informacje użyłem znacznika h3, przy czym dodałem też znacznik small wokół łącza edycji, aby było ono nieco mniejsze od znacznika czasu. Drugi element zagnieżdżony zawiera aktualną treść wpisu. Założona wcześniej estymata w sprawdziła się przypadku tego zadania.

7.4 Sprint Review/Demo

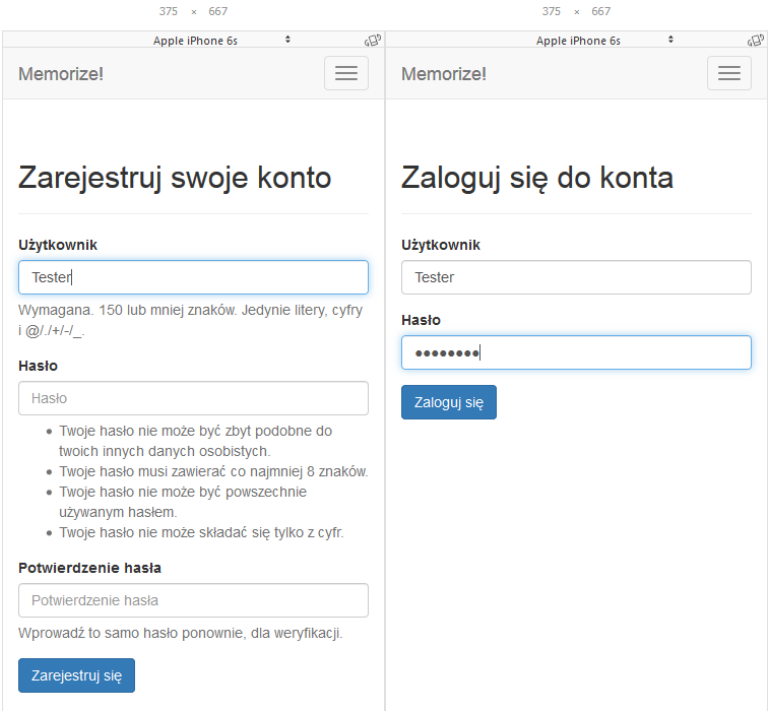
Trzeci sprint został zakończony powodzeniem. Wszystkie założone zadania zostały zrealizowane. Aplikacja stała się bardziej przejrzysta, charakteryzuje się spójnym stylem i jest czytelna. Od razu wiadomo do czego służy. Komunikaty mają styl spójny ze stroną i doskonale współgrają z ogólnym wyglądem naszej aplikacji. Demonstracja uzyskanego przyrostu:



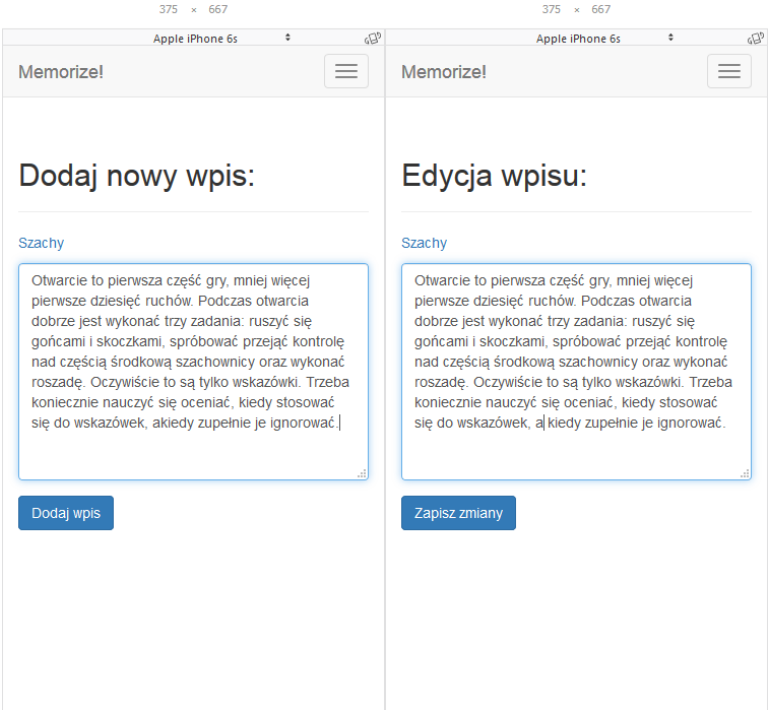
Tak prezentuje się strona główna naszej aplikacji w rozdzielczości 720p.



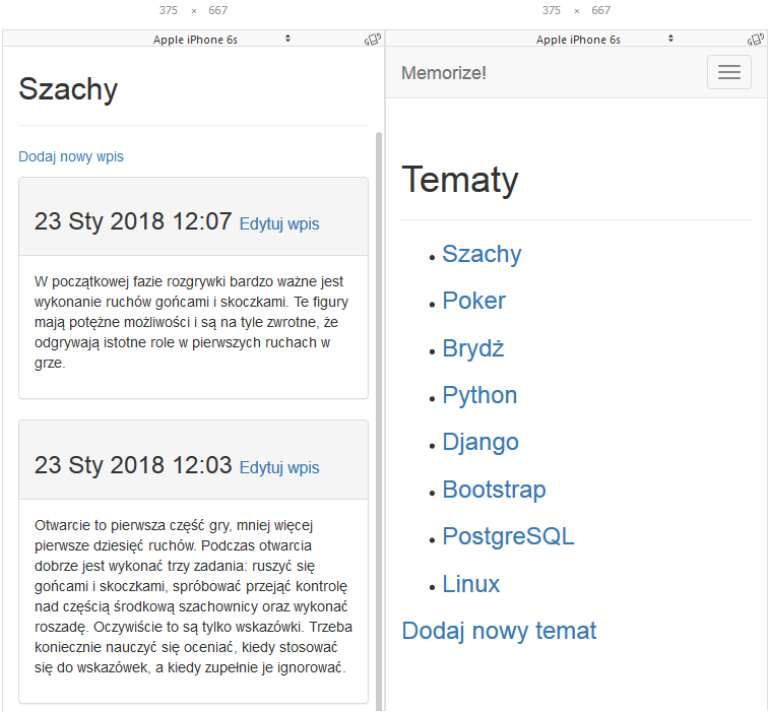
Tak prezentuje się strona główna naszej aplikacji na tablecie Apple iPad Air 2.



Ekrany rejestracji i logowania na smartfonie Apple iPhone 6s.



Dodawanie i późniejsza edycja wpisu na smartfonie Apple iPhone 6s.



Listy wpisów i tematów na smartfonie Apple iPhone 6s.

8 Sprint 4

8.1 Cel

Wdrożymy projekt na serwerze, aby każdy, kto ma dostęp do internetu mógł zarejestrować się i utworzyć konto w aplikacji.

8.2 Sprint Planning/Backlog

Tytuł zadania. Wdrożenie aplikacji "Memorize!".

- Estymata: "XL".

8.3 Realizacja

Tytuł zadania. Wdrożenie aplikacji "Memorize!".

Wykonawca. Zespół deweloperski.

Realizacja. Do wdrożenia naszej aplikacji wykorzystamy *Heroku*, platformę sieciową pozwalającą na zarządzanie operacją wdrażania aplikacji internetowej. Bezpłatne konto w *Heroku* ma pewne ograniczenia, ale nie są one dotkliwe i praktycznie bez żadnych kosztów możemy eksperymentować z wdrożeniem aplikacji. Konieczne jest zainstalowanie kilku dodatkowych pakietów, pomagających we wdrożeniu projektu na serwerze WWW. Pakiet *dj-database-url* pomaga Django w komunikacji z bazą danych używaną przez *Heroku*, a pakiety

static pomagają w prawidłowym zarządzaniu plikami statycznymi. Pliki statyczne zawierają reguły stylów oraz skrypty. Na początek *Heroku* musi wiedzieć jakie pakiety są niezbędne do działania naszego projektu. Zapisujemy w pliku *requirements.txt* nazwy wszystkich pakietów aktualnie zainstalowanych w projekcie. Upewniamy się, że platforma sieciowa będzie używać dokładnie takiej samej wersji Pythona, jakiej używaliśmy podczas pracy nad projektem. Na koniec przygotowujemy plik konfiguracyjny zawierający ustawienia dedykowane dla środowiska *Heroku*. Z poziomu aktywnej sesji powłoki przekazujemy pliki z repozytorium Gita do repozytorium utworzonego na *Heroku*. Uzyskujemy adres URL wdrożonego, ale jeszcze nie skonfigurowanego projektu. Konfigurujemy bazę danych w środowisku *Heroku*. Jako, że nie kopiowaliśmy danych lokalnych do serwera (co jest normalną praktyką), tworzymy nowe konto super - użytkownika.

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++)
...
```

».

8.4 Sprint Review/Demo

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://docs.python.org/3.5/`

[5] `https://docs.djangoproject.com/pl/1.11/`