Aplikacja webowa ToDoList

dokumentacja techniczna

Autorzy: Natalia Szyszka,
Aleksandra Niedziela

Spis treści

1. Opis projektu....3
2. Określenie wymagań....3
Określenie person....3
3. Architektura i zakres projektu....3
b) Mapa aplikacji....4
c) Tabelaryczna lista widoków z ogólnym opisem....4
d) Opis funkcjonalności na poszczególnych ekranach....4
e) Lista niezbędnych wymagań funkcjonalnych....6
a) Wersje językowe....6
4. Architektura technologiczna....6
5. Implementacja....6
6. Wymagania graficzne....10

1. Opis projektu

Celem aplikacji **ToDoList** jest pomoc w organizacji czasu. Użytkownik po zarejestrowaniu i zalogowaniu może tworzyć nowe zadania, które ma do zrealizowania w ciągu dnia oraz odhaczyć te, które już wykonał.

2. Określenie wymagań

Określenie person

ADAM NOWAK

Wiek	21 lat
Wykształcenie	średnie
Status	w związku
Praca	dorywcza
Inne	student

BIO

Mieszka w Poznaniu w wynajmowanym mieszkaniu. Planuje ukończyć studia i w międzyczasie pracować, by zarobić na utrzymanie (teraz dokładają mu rodzice).

HOBBY

- gry komputerowe
- dobry film
- sport

CELE

- ukończenie studiów z dobrym wynikiem
- osiągnięcie samodzielności finansowej
- znalezienie stałej pracy

SOCIAL MEDIA

- facebook
- instagram

FRUSTRACJE

- brak czasu (zła organizacja)
- zapominanie o obowiązkach

TECHNOLOGIA

- Internet
- social media
- online shopping

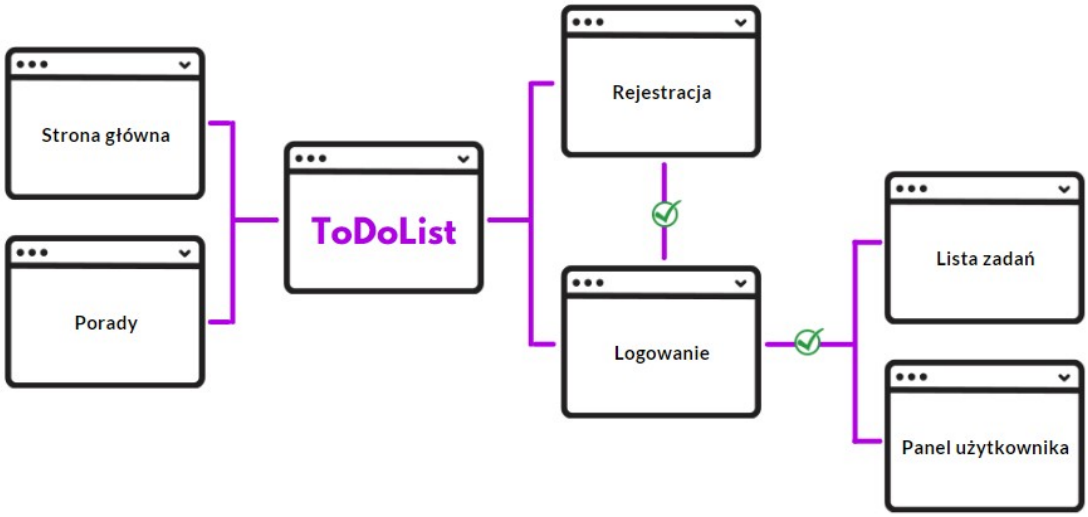
3. Architektura i zakres projektu

a) Architektura aplikacji.

Strona główna będzie głównym punktem wyjścia, będzie można z niej przejść na inne strony lub do panelu rejestracji bądź logowania; będzie zawierać informacje do czego służy aplikacja.

Po zalogowaniu użytkownik będzie miał możliwość przejścia do panelu użytkownika bądź listy zadań.

b) Mapa aplikacji.



c) Tabelaryczna lista widoków z ogólnym opisem.

Widok	**Opis**
Strona główna	Wyświetla informacje
Porady	Wyświetla informacje
Panel rejestracji	Wyświetla formularz
Panel logowania	Wyświetla formularz
Panel użytkownika	Wyświetla dane
Edycja danych użytkownika	Wyświetla formularz
Lista zadań	Wyświetla dane, możliwość zaznaczenia wykonania zadania
Dodaj zadanie	Wyświetla formularz
Edytuj zadanie	Wyświetla formularz
Usuń zadanie	Wyświetla prośbę potwierdzenia
Strona poprawnego wylogowania	Wyświetla informację

d) Opis funkcjonalności na poszczególnych ekranach.

- Strona główna

Wyświetla informacje na temat aplikacji. (dostępna publicznie)

- Porady

Strona wyświetlająca porady na temat tego jak lepiej zarządzać czasem. (dostępna publicznie)

- Menu

Umożliwia poruszanie się między stronami: strona główna i porady; a także przejście do panelu rejestracji i logowania. Po zalogowaniu zapewnia też dostęp do panelu użytkownika oraz listy zadań.

- Panel rejestracji

Umożliwia założenie konta użytkownika poprzez podanie adresu e-mail, hasła oraz nazwy nowego użytkownika w formularzu. Wysyła jego dane do bazy danych.

- Panel logowania

Łączy się z bazą danych użytkowników, umożliwiając użytkownikowi zalogowanie się na jego konto poprzez podanie adresu e-mail i hasła w formularzu. Po zalogowaniu użytkownik zyskuje dostęp do swoich danych (panelu użytkownika oraz listy zadań).

- Panel użytkownika

Do panelu użytkownika ma dostęp tylko zalogowana osoba. Umożliwia on zmianę danych użytkownika takich jak hasło i numer telefonu. Umożliwia również usunięcie konta. Panel wyświetla dane pobierając je z bazy danych oraz zapisuje w niej wszystkie zmiany.

- Lista zadań

Strona, do której dostęp ma wyłącznie zalogowany użytkownik. Może on za pośrednictwem formularzy w oddzielnych widokach dodawać zadania jakie ma do wykonania, edytować je, przeglądać szczegóły oraz usuwać. Użytkownik ma także możliwość zaznaczenia, że zadanie zostało wykonane. Może dodawać zadania zarówno przypisane do konkretnych godzin jak i zadania do wykonania w międzyczasie. Wszystkie dane są umieszczane w bazie danych. Strona pobiera dane z bazy danych w celu wyświetlenia listy zadań.

- Wylogowanie

Użytkownik ma możliwość wylogowania się. Po wylogowaniu wyświetlona zostaje strona główna aplikacji.

e) Lista niezbędnych wymagań funkcjonalnych.

- aplikacja musi posiadać bazę danych, łączyć się z nią, pobierać z niej dane i wstawiać do niej dane;
- aplikacja musi posiadać możliwość rejestracji kont użytkowników, logowania i wylogowania;
- aplikacja musi posiadać podstrony dostępne publicznie i podstrony tylko dla użytkowników;
- aplikacja musi uwzględniać indywidualną interakcję użytkownika (przesłanie formularza, z którego dane będą przechowywane w bazie i przypisane będą do tegoż użytkownika).

a) Wersje językowe.

- Polski

4. Architektura technologiczna

Aplikacja internetowa ASP.NET Core (Model-View-Controller)

Język programowania: C#, HTML, JavaScript

5. Implementacja

a) Użytkownicy

- W celu zrealizowania projektu została użyta paczka Microsoft ASP.NET Core Identity, przy pomocy której została rozwiązana kwestia rejestracji, logowania i wylogowywania użytkowników oraz przeglądanie przez każdego z nich własnych danych i ich edycja.
- Dzięki użytej paczce dane są walidowane, a hasła hashowane.

b) Strony: Strona główna oraz Porady

- Do zrealizowania obu stron został wykorzystany w głównej mierze język HTML wsparty biblioteką Bootstrap. Zdjęcia oraz tekst zostały zaimplementowane przy użyciu podstawowych tagów.

c) Lista zadań

- Każde zadanie jest oparte na modelu, który zawiera: id zadania, datę (w jaki dzień ma być wykonane), start i koniec (godziny wykonania zadania; podawane opcjonalnie), informację czy godziny wykonania zadania są istotne, nazwę, opis (opcjonalnie), informację czy zadanie zostało wykonane (domyślnie ustawione na fałsz), id użytkownika tworzącego zadanie (pobierane automatycznie podczas tworzenia/edycji zadania)
- Dzięki powyższym danym oraz poniższej linijce kodu, zastosowanej cztery razy w różnych konfiguracjach, zyskujemy widok naszej listy zadań. W górnej tabelce zadania do wykonania w konkretnych godzinach (pierwsze w kolejności są zadania do wykonania, potem już wykonane). W dolnej tabelce są zadania do wykonania w międzyczasie (pierwsze w kolejności są zadania do wykonania, potem już wykonane).

```
@foreach (var item in Model)
{
    <tr>
        @{
            if (item.UserId == ViewBag.Id && item.Hour == true && item.Done == false && item.Data.Date == ViewBag.Data)
            {
```

Sprawdzamy czy id użytkownika się zgadza z id użytkownika przypisanego do zadania, czy istotna jest godzina wykonania zadania, czy zadanie zostało wykonane, sprawdzamy też datę dnia, dla którego użytkownik chce wyświetlić zadania.

- Formularz wyboru dnia z którego zadania mają być wyświetlone (dostępny jest nad tabelami).

```
Wybierz dzień:
@using (Html.BeginForm())
{

    <input type="date" name="data" value="@Convert.ToDateTime(ViewBag.Data).ToString("yyyy-MM-dd")" />
        <button class="button">Prześlij</button>
}
```

Kod z widoku

- Jeśli nie wybierzemy daty, automatycznie wyświetla dzień dzisiejszy.

```
public async Task<IActionResult> Index(DateTime? data)
{
    var userDbContext = _context.ModelZadania.Include(m => m.User);


    ViewBag.Data = data;

    var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
    ViewBag.Id = userId;

    if (data == null)
    {
        ViewBag.Data = DateTime.Today;
    }

    return View(await userDbContext.ToListAsync());
}
```

Kod z kontrolera

- Poniższy kod przy każdym zadaniu umieszcza przyciski, które przenoszą do odpowiednich podstron (Edytuj, Szczegóły, Usuń), jeden z nich umożliwia też zaznaczenie czy zadanie jest wykonane czy jeszcze nie (jednocześnie sam na sobie wyświetla informację czy zadanie jest „Gotowe” czy „Do zrobienia”).

```
<td>
    <button type="button" class="button button4" onclick="location.href='@Url.Action("Done", "ModelZadanias", new { id=item.ZadanieId })'">Do zrobienia</button>
</td>
<td>
    <button type="button" class="button" onclick="location.href='@Url.Action("Edit", "ModelZadanias", new { id=item.ZadanieId })'">Edytuj</button>
    <button type="button" class="button" onclick="location.href='@Url.Action("Details", "ModelZadanias", new { id=item.ZadanieId })'">Szczegóły</button>
    <button type="button" class="button" onclick="location.href='@Url.Action("Delete", "ModelZadanias", new { id=item.ZadanieId })'">Usuń</button>
```

Kod z widoku

- Opcje Edytuj, Szczegóły, Usuń są dostępne tylko dla właściciela zadania.

```
var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
ViewBag.Id = userId;
```

Kod z kontrolera. Każda opcja zawiera poniższą linijkę tak, aby można było pobrać i przechować id zalogowanego użytkownika, a następnie porównać z id użytkownika, który utworzył zadanie.

- Jeśli id zalogowanego użytkownika i id użytkownika, który stworzył zadanie są inne (np. przy próbie edycji nieswojego zadania za pośrednictwem linku) zostaje wyświetlony komunikat o próbie oszustwa.

```
@{
    if (Model.UserId != ViewBag.Id || Model.UserId == null)
    {
        <h1><span style="color:red ;"><b>Oszukista!</b></span></h1>
        <h3>Dlaczego próbujesz mnie oszukać?! To zadanie nie należy do Ciebie!!!</h3>
        <br>
        <img src="~/img/zly.png" style="max-height:250px" alt="zly" />
    }
}
```

Kod z widoków dla opcji Edytuj, Szczegóły, Usuń

d) Kontrolery

(1) HomeController

- wyświetlający metody służące do wyświetlenia Strony głównej oraz strony Porady

(2) ModelZadaniasController zawiera metody:

- Index – umożliwia wyświetlenie listy zadań.
- Create – umożliwia dodanie nowego zadanie.
- Edit – umożliwia edycję zadania.
- Details – umożliwia wyświetlenie szczegółów zadania.
- Delete – umożliwia usunięcie zadania.
- Done – umożliwia zaznaczenie czy zadanie jest *Gotowe* czy *Do zrobienia*.

6. Wymagania graficzne

- Aplikacja jest przejrzysta dla wszystkich użytkowników, zarówno zalogowanych jak i niezalogowanych. Każdy w prosty sposób może odnaleźć to co potrzebuje.
- Estetyczny interface.
- Kolory aplikacji: fioletowy, biały, szary.