Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра Информатики

Дисциплина: Программирование

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

Веб-сервис для заметок

Выполнил: студент гр. 653501

Медунецкий М.А.

Руководитель: ассистент кафедры информатики Козуб В.Н.

Минск 2017

**Оглавление**

Введение [3](https://docs.google.com/document/d/1sOqUUBWZ36Ob7O4Nt_wgEFUHMKzEecCZiYJN2b_9WhI/edit#heading=h.30j0zll)

Анализ предметной области4

Постановка задачи4

Теоретическая часть4

Исследование аналогов4

Разработка программы5

Разработка серверной части5

Разработка клиентской части9

Работа программы:11

Заключение15

Список литературы:16

**Введение**

В последнее время интернет стал неотъемлемой частью нашей жизни, растет популярность социальных сетей и различных медиа-сервисов. Еще с давних времен люди делали заметки в блокнотах, тетрадях.

Так как носить с собой блокнот неудобно, а интернет есть практически везде, то реализация веб-сервиса заметок является актуальным.

**Анализ предметной области**

*Постановка задачи:*

Целью курсовой работы является создание веб-сервиса, со следующим функционалом

1. Создавать, редактировать и удалять заметки.
2. Отображать заметки конкретного пользователя.
3. Просматривать и редактировать информацию пользователя.

*Теоретическая часть*

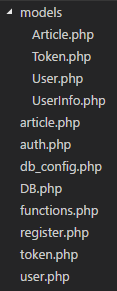
Для создания курсового проекта использовался фреймворк «Angular». Причинами выбора данного фреймворка стали:

1. Использование TypeScript. Это даёт некоторые преимущества перед стандартным JavaScript, например классы и интерфейсы.
2. Встроенные директивы повышают удобство работы с html
3. Архитектура приложения на основе компонентов, что дает расширяемость.

*Исследование аналогов*

При исследовании аналогов оказалось, что в большинстве случаев такие сервисы обладают слишком большим функционалом, что затрудняет их использование. Этот недостаток и стал одной из причин создания этого курсового проекта.

**Разработка программы**

**Разработка серверной части**

Рассмотрим структуру серверной части, которая написана на PHP. Все файлы можно разделить на 4 группы:

1. Модели
2. Скрипты для работы с клиентом (интерфейсы)
3. Скрипты для работы с базой данных
4. Набор вспомогательных функций

Рассмотрим каждую группу в отдельности.

*Модели*

Модели в данной программе содержат класс объектов с которыми будут взаимодействовать скрипты программы. Например, сохранять их в базу данных. Всего в программе 4 модели:

1. Заметка
2. Токен
3. Пользователь
4. Информация о пользователе

В основном модели содержат набор полей. Их код можно посмотреть на гитхабе.

*Работа с базой данных*

Операции с базой данных выполняет класс DB, описанный в файле DB.php. Он содержит 4 публичных метода: add, find, update, delete - которые выполняют соответствующие операции. Этот класс занимается созданием SQL-запросов, и общением с базой данных, что упрощает разработку приложения, ведь в дальнейшем я оперирую объектами и не думаю, как сохранить или достать этот объект из БД.

Рассмотрим метод add подробнее. Вначале происходит проверка, является ли аргумент массивом, если нет, то вызываем метод add\_one, иначе вызываем add для каждого элемента массива.

Код метода add\_one:

private function add\_one($object)

{

$table = get\_class($object).'s';

$key\_query = '';

$val\_query = '';

$comma = '';

foreach ($object as $key => $value)

{

$key\_query .= $comma."`$key`";

$val = $this->process\_value($value);

$val\_query .= $comma."$val";

$comma = ',';

}

$q = "INSERT INTO `$table` ($key\_query) VALUES ($val\_query);";

$result = $this->db->query($q);

return $result;

}

Функция get\_class возвращает название класса, к которому добавляется английская буква ‘s’. Полученное значение является названием таблицы. Например: класс Article - таблица Articles, класс User - таблица Users.

Код этого класса можно посмотреть на гитхабе.

*Работа с клиентом*

Клиент взаимодействует с сервером посредством четырех типов запросов (REST), в запросе может содержаться заголовок Authorization, который содержит токен пользователя и по которому сервер узнает идентификационный номер. Зная ID сервер разрешает или запрещает выполнение операции и посылает ответ, по-статус коду которого можно понять состояние операции. Например, 200 - успешно, 401- не авторизирован и т. д.

Скрипты взаимодействия:

1. Article.php
2. Register.php
3. Token.php
4. User.php

Рассмотрим процесс взаимодействия подробнее на примере article.php.

При получении запроса происходит проверка на тип запроса и в зависимости от типа выбирается метод, который будет работать вдальнейшем.

Если для выполнения метода требуется авторизация, то вызывается следующая функция:

function auth\_required()

{

global $user\_id;

if (!$user\_id)

{

http\_response\_code(401);

die();

}

}

Пример функции, которая обрабатывает DELETE запрос:

function delete\_post()

{

global $db, $user\_id;

auth\_required();

$post = new Article();

$post->id = $\_GET['id'];

$post->user\_id = $user\_id;

if (!$db->delete($post))

{

http\_response\_code(400);

}

}

**Разработка клиентской части**

Клиентская часть написана на фреймворке Angular. Приложение состоит из компонентов, сервисов, моделей и главного модуля приложения.

*Модели*

Модели содержат классы, которые используются сервисами и компонентами. В проекте 3 модели:

1. Article - содержит номер статьи, её заголовок, содержание и ID автора
2. User - Содержит ID пользователя, его логин, информацию о нем и номера статей.
3. UserAdditionalInfo - представляет ключ и значение. Например ключ - avatar, значение - ссылка на него.

*Компоненты*

Компонент встраивает html-код в DOM, он может быть вызван напрямую, например <app-header></app-header>, либо через router-outlet, который встроит компонент, в зависимости от URL. В приложении представлено 13 компонентов:

1. AppHeader - шапка приложения
2. AppMenu - меню сбоку
3. App - главный компонент, подгружает остальные через router-outlet
4. ArticleBlock - просмотр заметки
5. ArticleCreator - компонент для создания заметки
6. ArticleEditor - компонент для редактирования заметки
7. ArticleViewer - компонент для просмотра заметки, В отличие от ArticleBlock, выводит полную информацию.
8. ArticlesGroup - выводит заметки, номера которых были переданы в параметр articlesId
9. Auth - форма авторизации и регистрации
10. DefaultTemplate - шаблон, который применяется к болшинству страниц
11. NotFound - отображается, когда ничего не было найдено
12. Settings - работа с настройками пользователя
13. UserProfile - отображает информацию о пользователе и его статьи.

Код всех компонентов, как и весь проект можно посмотреть на гитхабе.

*Сервисы*

В проекте 5 сервисов, которые взаимодействуют друг с другом:

1. Article - содержит методы для работы с заметкой
2. Auth - содержит методы для регистрации, авторизации, проверки на авторизацию, получение ID пользователя
3. Request - содержит 4 метода, которые отправляют GET, POST, UPDATE, DELETE запросы соответственно
4. Storage - сервис для чтения и записи данных в браузерное хранилище
5. User - сервис для получения модели пользователя по ID, а также изменения информации текущего пользователя

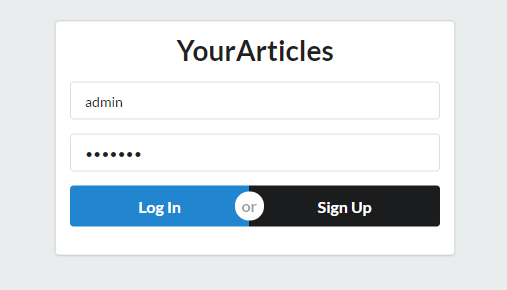
*Главный модуль*

Главный модуль описывает структуру приложения, роуты, сервисы, компоненты и заимствованные модули.

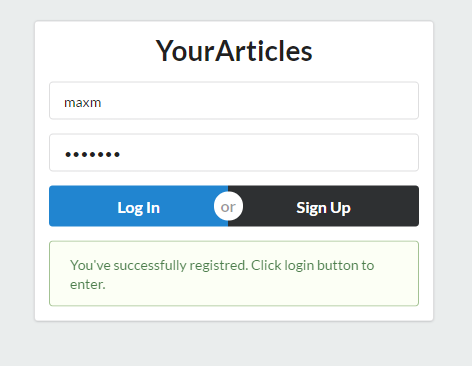


**Работа программы**

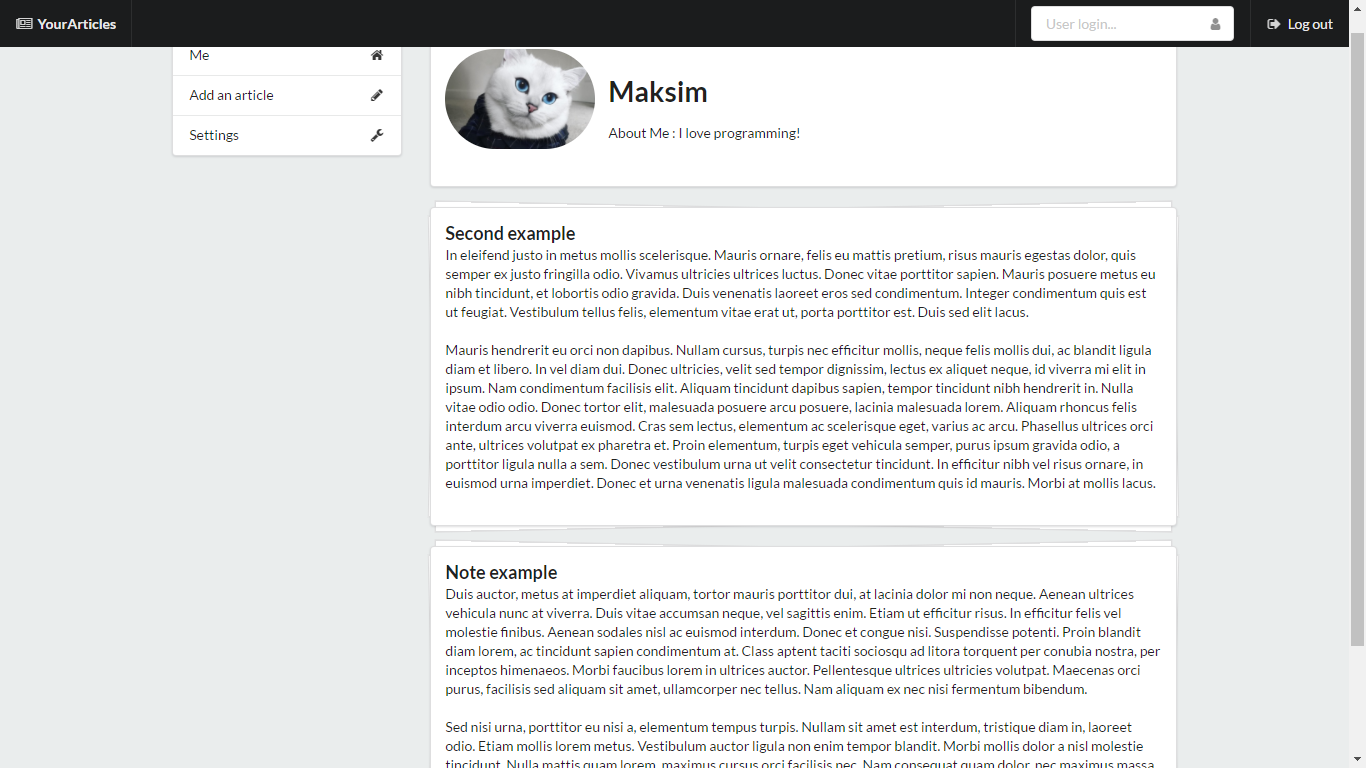
Форма входа выглядит следующим образом:



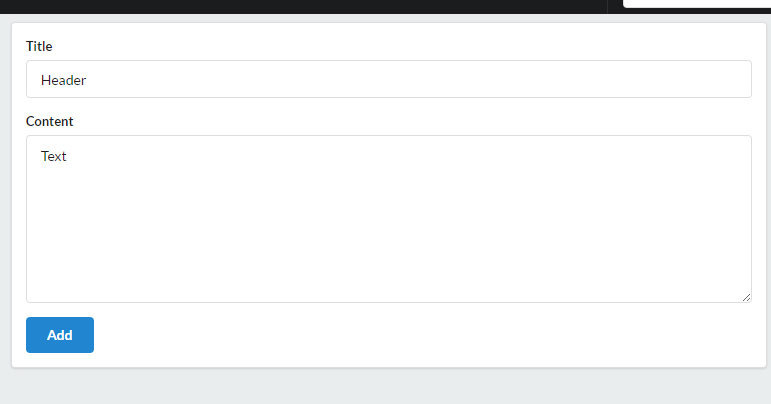
На этой же странице пользователь может зарегистрироваться:



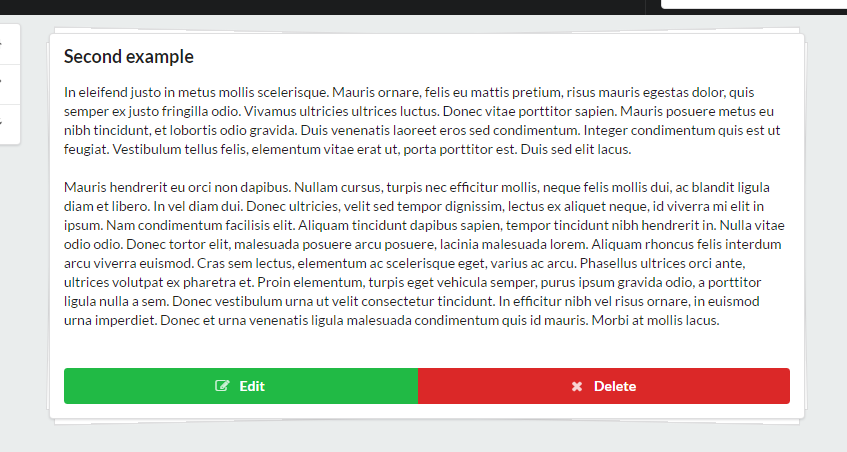
Так выглядит профиль пользователя:



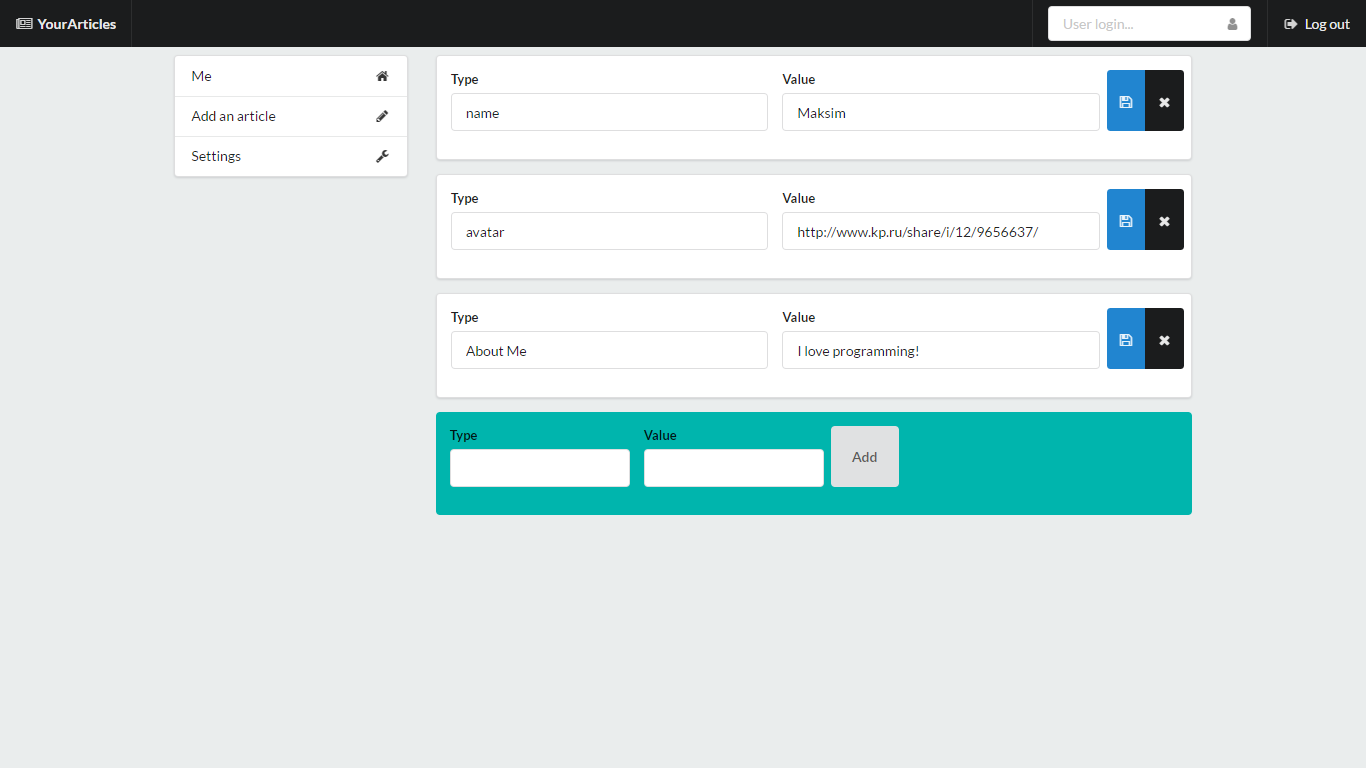
Редактор заметок:



Страница просмотра заметки. Если пользователь её автор, то будут показаны кнопки редактирования и удаления.



Страница настроек.



**Заключение**

При выполнении курсовой работы были выполнены следующие задачи:

- Был освоен фреймворк «Angular».

- Были освоены языки программирования TypeScript и PHP.

- Был написан веб-сервис заметок, в следствие чего повысился навык в написании веб-приложений типа SPA (Single Page Application). Также написание проекта было устроено таким образом, что возможна его модификация и дополнение не затрагивая уже написанный код. Также организация хранения информации об пользователях устроена абстрактно, что позволяет модифицировать приложение под другие базы данных, без переписывания всего кода.

**Список литературы**

1. ng-book, Vijay Ganta - https://www.ng-book.com/2n/
2. PHP Manual - https://secure.php.net/manual/en/