**Компьютерная академия «ШАГ»**

**Днепропетровский филиал**

**Кафедра Разработки программного обеспечения**

**ДИПЛОМНЫЙ ПРОЕКТ**

**“Пробное тестирование «ЗНО» "**

**Студента группы ЕКО-15-П-1  
Быкова Н.К.**

**Научный руководитель:  
Шаптала М. В.**

**Содержание**

**Введение 3**

**1. Постановка задачи 5**

**2. Требования к функциональным характеристикам системы 7**

**3. Требования к техническим характеристикам 9**

**4. Требования к программным характеристикам 10**

**5. Проектирование системы 11**

1. **Проектирование интерфейса……………………………………..……11**
2. **Проектирование диаграммы классов …………………...…………...13**
3. **Проектирование базы данных …………………………………...........14**
4. **Проектирование алгоритмов ………………………………….............19**

**6. Руководство пользователя ………………………………............................19**

**7. Список литературы …………………………………………………………24**

**Введение**

Множество сайтов со всего мира используют сервера для хранения и обработки свой служебной информации. Эта информация хранится в так называемых базах данных.

**База данных** - это хранилище для большого количества систематизированных данных, с которыми можно производить определенные действия (добавление, удаление, изменение, упорядочивание и прочее).

Все данные, находящиеся в базе данных, можно представить в виде записей или объектов.

Для успешной работы с базами данных нужны программные средства, которые обеспечивали бы доступ к нужной информации, внесение каких-либо изменений в базу данных и другие действия с данными. Для решения этой задачи используются системы управления базами данных.

**Система управления базами данных (СУБД)** - это совокупность языковых и программных средств, обеспечивающих создание, использование и ведение базы данных.

Все СУБД делятся на две группы:

* *локальные;*
* *сетевые.*

**Локальные** - это СУБД, работающие на одном компьютере. К ним относятся sqlite, Microsoft Access и т. д.

**Сетевые** - это СУБД, позволяющие нескольким компьютерам использовать одну и ту же базу данных с помощью технологии клиент-сервер.

В дипломном проекте в качестве СУБД был выбран My SQL Server - пожалуй, один из самых быстрых и удобных в разработке систем управления базой данных. Широкое применение во многих сферах, начиная с мелких предприятий и заканчивая огромными корпорациями, лишь подтверждает качество и надежность сервера. Постоянная поддержка и выход новых версий позволяют идти в ногу со временем.

Для доступа к данным хранящимся на SQL-сервере необходимо разработать REST API для платформы .NET Core. Основным языком программирования для платформы .NET Core является высокоуровневый объектно-ориентированный язык C#, который и был использован в курсовом проекте.

**С#** - объектно-ориентированный язык программирования разработанный как основной язык разработки приложений для платформы **Microsoft.NET**. Среда программирования **Microsoft Visual Studio 2017** с поддержкой **С#** предоставляет все необходимые инструменты для создания приложений, работающих с базой данных.

**.NET Core** основана на [**.NET Framework**](https://ru.wikipedia.org/wiki/.NET_Framework). Платформа **.NET Core** отличается от неё модульностью, кроссплатформенностью, возможностью применения облачных технологий, и тем, что в ней произошло разделение между библиотекой **CoreFX** и средой выполнения **CoreCLR.**

**Entity Framework** является продолжением технологии Microsoft **ActiveX** Data и предоставляет возможность работы с базами данных через объектно-ориентированный код **C#**. Этот подход предоставляет ряд существенных преимуществ: вам не нужно беспокоиться о коде доступа к данным, вам не нужно знать деталей работы **СУБД SQL Server** и синтаксиса языка запросов **MySql**, вместо этого вы работаете с таблицами базы данных как с классами C#, с полями этих таблиц - как со свойствами классов, а синтаксис SQL-запросов, который в ADO.NET раньше нужно было вставлять в код C# в виде команд, заменен на более удобный подход с **LINQ**. Entity Framework берет на себя обязанности по преобразованию кода C# в SQL-инструкции.

Но кроме обработки и манипуляции данных, их надо отображать и должна быть выбрана технология подходящая для этих нужд.

«PHP: [препроцессор](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80) [гипертекста](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%BF%D0%B5%D1%80%D1%82%D0%B5%D0%BA%D1%81%D1%82)»; первоначально *Personal Home Page Tools* — «Инструменты для создания персональных веб-страниц») — [скриптовый язык](https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством [хостинг-провайдеров](https://ru.wikipedia.org/wiki/%D0%A5%D0%BE%D1%81%D1%82%D0%B8%D0%BD%D0%B3) и является одним из лидеров среди языков, применяющихся для создания [динамических веб-сайтов](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%81%D0%B0%D0%B9%D1%82).

1. **Постановка задачи**

Темой дипломной работы является создание системы автоматического оценивания знаний студентов.

Существует большое количество различных систем оценивания знаний студента. Это как и автоматические, так и ручные (преподаватель или администратор) сам выставляет оценки. Однако они тяжело масштабируемы, если необходимо ввести более сложные вопросы – например с картинками или видео вопросы.

Был взят для примера тести с зовнішнього незалежного тестування. В нем имеется большой выбор предметов и различный категорий и вариантов вопросов, также и ответов на них.

Система обязуется избавить преподавателей от проверки вопросов, где есть варианты и конкретные ответы. Преподаватель лишь должен проверять вопросы, где студенту необходимо изложить полный ответ на вопрос.

Благодаря тому что был выбран внешний ресурс для формирования тестов, вручную заносить все вопросы, представленные на сайте, заняло бы много времени. Потому кроме базового приложения, необходимо реализовать парсинг сайта – zno.osvita.ua. Это не только увеличит скорость наполнения и разворачивания базы данных, но и дальнейшее выгрузка новых внедренных тестов.

Система не должна быть привязана только к конкретному сайту.

В дальнейшем, ее можно перенести на мобильное и дэсктопное приложения. Именно в таких ситуациях и применяется написание REST API. Это позволит нескольким приложениям использовать одну единую базу тестирования.

Особенно важно подчеркнуть, что api должно быть не только высокопроизводительным, но и переносимым между различными серверами.

Целью дипломной работы является:

- Подчеркнуть знания в написании масштабируемых проэктов.

- Закрепить навыки в использовании скриптового языка, написания веб – приложения на PHP

- Закрепить знание в написании приложение на .Net Core.

- Закрепить знание в использовании систем управления версий (например Git).

**2. Требования к функциональным характеристикам системы**

В процессе дипломного проекта необходимо разработать систему автоматического тестирования.

Система должна проверять знания студента, на предопреденных вопросах с ответами. Преподаватель же должен проверять только вопросы с расширенными ответами.

Программная часть должна обеспечивать возможность выполнения перечисленных ниже функций:

1. Неавторизованный доступ:
   1. На главной странице отображается форма регистрации/авторизации
2. Регистрация (роль - пользователь)
   1. При регистрации вводится телефон (обязательно - валидация), пароль (обязательно), email (опционально - валидация).
   2. После регистрации сразу (без подтверждения) переходим к странице тестирования
3. Авторизованный доступ
   1. С ролью пользователя
   2. С ролью преподавателя
      1. Регистрацию преподавателя выполняет администратор
   3. С ролью администратора

**Тестирование**:

1. **Пользователь** после регистрации/авторизации переходит на страницу тестирования автоматически
2. На странице тестирование присутствует кнопка “Начать тест”, нажав на которую начинается тестирование. При этом на странице отображается таймер с обратным отсчетом времени.
3. Вопросы выводятся по одному на страницу.
4. Ответы на вопросы сохраняются на сервере автоматически (как только дан ответ).
5. Во время тестирования кроме вопроса отображаются кнопки: следующий вопрос, предыдущий вопрос, завершить тест.
6. Таймер продолжает счет даже если была прервана сессия (при закрытии браузера).
7. После истечения заданного времени выводится сообщение “Время вышло”.
8. При нажатии на кнопку “Завершить тест” отображается сообщение “Тестирование завершено”.
9. При наличии email у пользователя и отсутствии в тесте вопросов с задачей (которые должен проверять преподаватель) - результаты теста автоматически направляются на указанный email при регистрации.
10. В результатах указывается вопрос, ответ пользователя, правильный ответ. В конце выводится статистика - количество верных/неверных ответов и оценка.
11. При наличии вопросов с задачей тест должен быть проверен преподавателем (не первоочередная задача)

**Администрирование**

1. **Администратор** задает текущий тест (математика или физика или …)
2. Определяет вопросы каких годов и каких категорий сложности будут выдаваться пользователям
3. Определяет количество вопросов каждой категории сложности
4. Может задать время тестирования, по умолчанию:

Английский язык 150 минут;

Математика: 180 мин;

Украинский язык и литература: 180 минут;

Физика: 150 мин.

1. Вопросы должны выдаваться в случайном порядке в рамках данной категории
2. Может видеть статистику по пользователям: сколько сейчас тестируются, сколько завершили тест.
3. Может видеть результаты по каждому пользователю после завершения теста.
4. Может завершить тест преждевременно вместо пользователя.
5. Регистрирует преподавателя, который будет проверять тест
6. Указывает одного или нескольких преподавателей, которые будут проверять тесты (для вопросов с задачами) - не первоочередная задача

**Преподаватель**

1. Авторизуется в системе
2. После авторизации переходит к странице со списком тестов, которые необходимо проверить. Преподаватель не видит данных пользователя т.к. проверка должна быть анонимной

**3. Требования к техническим характеристикам**

Минимальные требования для работы программного обеспечения необходимо, что бы используемые вами технические средства соответствовали ниже перечисленным характеристикам:

1. Backend:

Процессор: 1 x Intel Xeon D-1521 @ 2.40GHz;

Доступная RAM: 2048 MБ;

Дисковое пространство: 10 ГБ;

Передача данных: без лимита;

Полный контроль с аккаунтом root: да

Доступ к терминалу в панели: да

Доступ через SSH: да

Пропускная способность: 1 ГБ/сек.

1. Frontend:

Дисковое пространство: 1 ГБ;

Передача данных: 5 ГБ/месяц;

Максимальный размер файла: 10 MБ;

**4. Требования к программным характеристикам**

Для эксплуатации данного программного обеспечения необходимо, что бы на компьютере были установлены:

* MySQL Server;
* Apache;
* Подключение к сети Интернет.

**5. Проектирование системы**

**5.1 Проектирование интерфейса**

В качестве технологии разработки Frontend части приложения, было решено выбрать PHP. Framework для взаимодействия сперва был выбран Yii2, но поскольку взаимодействие с базой данных происходит через внешнее Api и базовые библиотеки имеют внушительный размер на дисковом пространстве, было решено заменить это на самописный framework, основанный на паттерне MVC.

Шаблон, для отображения и манипуляции с данными был выбран AdminLTE-2.3.0 - <https://github.com/mybigjsdream/AdminLTE-2.3.0.git>;

Он имеет достаточное количество компонентов и шаблонов не только для данной стадии, но и для дальнейшего масштабирования.

Было реализовано 2 разных шаблона:

* Для администратора и преподавателя
* Для пользователя

В данной реализации они оба используют AdminLTE-2.3.0.

Часть запросов на api, выполняется не только на стороне JavaScript скриптов, но и на стороне PHP. Это сделано с целью дополнительного распределения ролей на контроллерах.

Благодаря тому, что вопросы были взяты с сайта zno.osvita.ua – расположения и шаблон вопросов был взят оттуда.

У администратора есть следующие страницы:

* Тесты
* Вопросы
* Уровни сложности
* Настройка теста

В дальнейшем предполагается:

* Страница с активными пользователями (кто проходит и уже прошел тест)
* Настройка множества тестов и по многим предметам
* Регистрация и отображение преподавателей

У пользователя:

* Страница с началом теста
* Страница тестирования

В дальнейшем предполагается:

- Осуществлять переход по вопросам

В данный момент преподаватель не может заходить на сайт, так как в системе возможен лишь отображение вопросов с одним правильным ответом (подробнее о вопросах в пункте **Проектирование базы данных и алгоритмов**). Для разделения вопросов, в проекте представлены «виджеты» с конкретным типом вопроса, подключаемые к страницам.

.

**5.2 Проектирование диаграммы классов**

**1) Backend**

**1.1 Zno.DAL**

**Abstration**

**IGenericRepository** – Обобщенный репозиторий, осуществляющий поиск в базе данных определенную сущность

**IUnitOfWork** – репозиторий, для поиска в таблицах

**IUserRepository** – репозиторий таблицы пользователей

**Entities**

**Answer –** сущность вопроса

**AnswerType –** тип вопроса

**ApplicationUser –** пользователь (админ, преподаватель, пользователь)

**ContentType –** тип контента (строка, картинка, видео)

**GeneratedTest –** сгенерированный тест, когда есть настройки теста

**Question** - вопрос

**Subject -** предмет

**Test -** тест

**TestSettings –** настройки теста

**TestSettingsAnswerType –** связь типа вопроса с настройками

**TestType –** тип теста

**UserAnswer –** ответы пользователя

**Implementation –** конкретные реализации сущностей от интерфейса **IGenericRepository**

**1.2 Zno.Parser**

**QuestionBodyAnswerParseImpl –** парсинг ответов

**FactoryAnswerParser –** в зависимости от типа вопроса, парсит конкретный вопрос и ответ (один, множество, задача, расширенный ответ)

**QuestionParseImpl –** парсинг тела вопроса

**IFrameQuestionBody –** парсинг видео

**ImageQuestionBody –** парсинг картинки

**SerializeQuestionBody –** сериализация тела вопроса в файл

**TextQuestionBody -** парсинг простого текста

**1.3 Zno.Server**

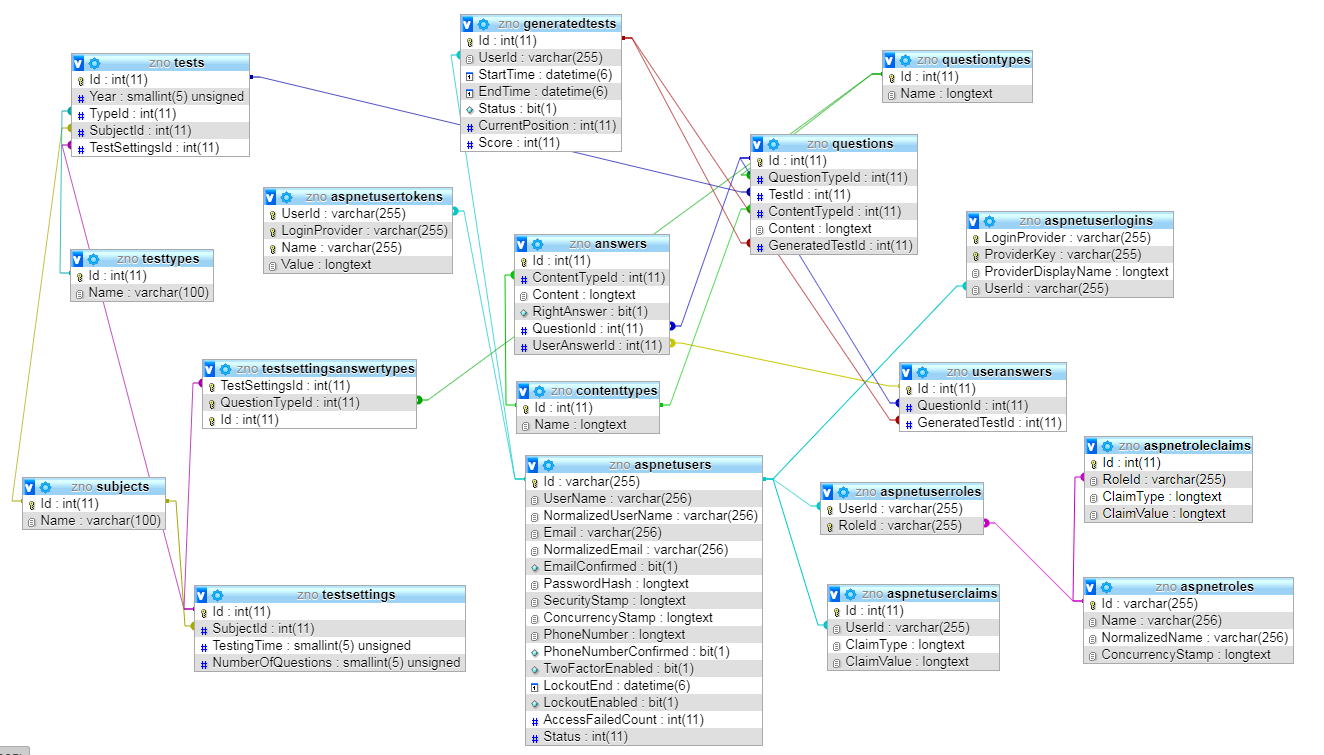
Controllers

AccountController – авторизаци, регистрации

TestController – настройка теста

TestingController - прохождение, начало, завершение теста

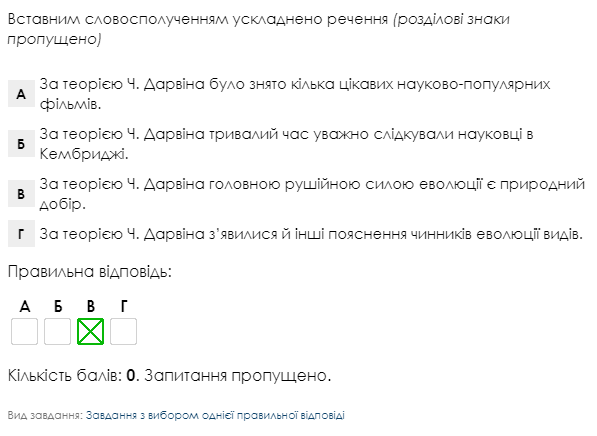
**5.3 Проектирование базы данных**



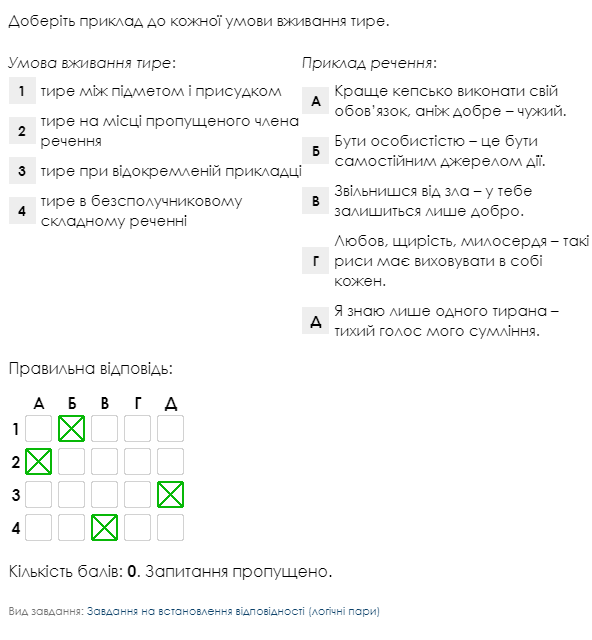
Из-за сложной структуры вопросов, представленной на zno.osvita.ua, структурированную таблицу создать не вышло.

Как упоминалось выше, вопросы и ответы на сайте zno.osvita.ua имеют следующий вид:

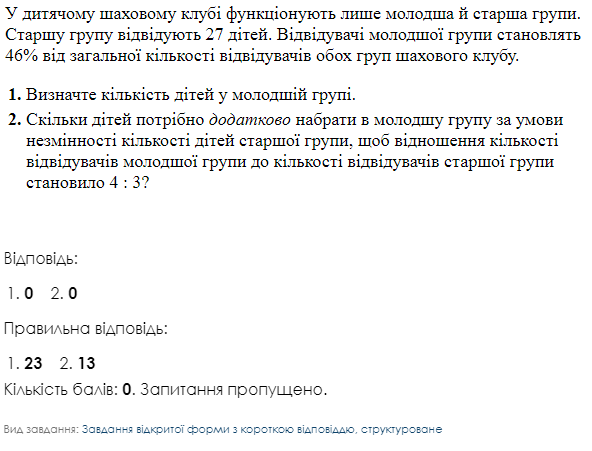
1. С одним правильным ответом



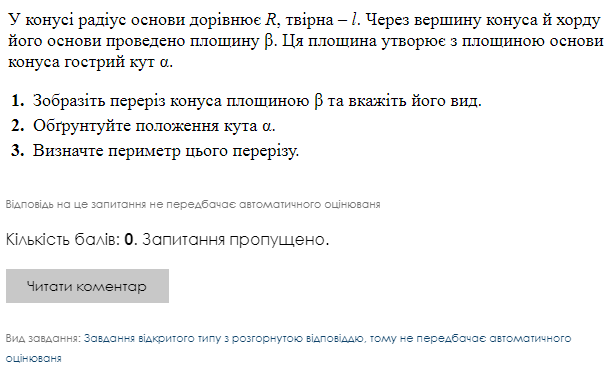
1. Множество правильных ответов



1. Расширенный ответ



1. Расширенный ответ с проверкой преподавателя



Вопросы и ответы могут хранить как в:

* Текстовом
* Изображение
* Видео

Потому в таблицу с вопросом и ответом, есть внешняя таблица ContentType, характеризующая определенный тип контента.

Сам контент вопроса и ответа – это JSON объект, имеющий следующую структуру – {“Url”: “”, “Title” : “”, “ContentType”:””};

ContentType – тип контента вопроса или ответа, применятся, поскольку в ответе также может быть текст или картинка

Url – содержит путь к картинке или видео

Title – текстовое содержимое.

Контент ответа с множественными правильными ответами, имеет следующую структуру – {“FirstBlock”:””, “TwiceBlock”:””};

Если посмотреть на скриншот с множественным выбором, можно увидеть, что тут необходимо найти сопоставление в двух блоках. Если на примере с одиночным или расширенным, указывается поле RightAnswer над правильным ответом и false над неправильным , то с множественным, указывается первый строка, со вторым и находится сопоставления по 4 ответам. Пятый же, в таком случае, записывается как неправильный, и поле FirstBlock не задается, только TwiceBlock.

**5.4. Проектирование алгоритмов**

Выше, было описано как хранятся данные о вопросе и ответе в базе. Теперь рассмотрим как они записываются.

В Backend проекте, есть библиотека ZnoParser. В конструкторе класса ZnoParser задаются базовые предметы (украинский язык, математика, физика и английский язык). При вызове метода StartParsing, производить заход на главные страницы предметов, где осуществляется парсинг заголовков всех тестов в предмете. Как только есть ссылка на конкретный тест, осуществляется запрос на их api, где приходит json, в который вложен html с правильными ответами на все вопросы.

Поскольку контент ответа и вопроса может быть разный, был реализован паттерн стратегия, для определения контента и соответствующий парсинг.

Пример парсинга ответов, если контент картинка:

private void \_ParseByNoneTableAnswers(HtmlNode taskCard, HtmlNode answers)

{

// Получаем tr теги с правильными ответами

var trRightAnswers = taskCard.EndNode.SelectSingleNode(XPathBuild.XPathFromNode(taskCard, "//table[@class=\"select-answers-variants\"]//tr[2]"));

if (trRightAnswers != null)

{

int idx = 0;

foreach (var td in trRightAnswers.ChildNodes)

{

// Проверка на наличие классов marker ok в блоке ответов

HtmlAnswer htmlAnswer = new HtmlAnswer();

var markerOk = td.EndNode.SelectSingleNode(XPathBuild.XPathFromNode(td, "//\*[@class=\"marker ok\"]"));

htmlAnswer.Content = (idx + 1).ToString();

htmlAnswer.HtmlContentType = Enums.HtmlContentType.String;

htmlAnswer.IsRight = markerOk != null;

htmlAnswers.Add(htmlAnswer);

idx++;

}

}

}

Для удобного разбора html, задействована библиотека HtmlAgilityPack.

Разбор предметов и тестов в нем происходит поочередно, по мере прохождения по дереву узлов.

Для того чтобы пользователь смог проходить тест, задается настройка теста.

Она включает в себя – затраченное время на тест, количество вопросов с одного типа вопроса на 1 тест, сами выбранные тесты и типы вопросов.

После этого, пользователь отправляет запрос на следующий action контроллера Testing

/// <summary>

/// Генерирование нового теста по указанному предмету

/// </summary>

/// <param name="subjectId">Индентификатор предмета по которому пользователь хочет пройти тест</param>

/// <returns></returns>

[Authorize(Roles = "User")]

[HttpPost]

public async Task<IActionResult> CreateNewTestV2(int subjectId)

Система проверяет есть ли у этого пользователя не пройденные тесты, если нет, то добавляет новый.

Каждый новый ответ сохраняется в action

/// <summary>

/// Принятие ответа на указанный вопрос указанного теста

/// </summary>

/// <param name="questionId">Индентификатор вопроса на который текущий пользователь отвечает</param>

/// <param name="answers">Ответы на вопрос</param>

/// <param name="generatedTestId">Идентификатор сгенерированного теста</param>

/// <returns></returns>

[Authorize(Roles = "User")]

[HttpPost]

public async Task<IActionResult> SavingAnswer(int questionId, [FromBody]List<Answer> answers, int generatedTestId)

Как только произвелись ответы на все вопросы – пользователь отправляет запрос на

/// <summary>

/// Завершение теста и подсчет результата (без учёта ответов типа Task)

/// (в процессе реализации, ошибка с получением ответов в GeneratedTest)

/// </summary>

/// <param name="generatedTestId">Идентификатор сгенерированного теста</param>

/// <returns>Сумма баллов за тест</returns>

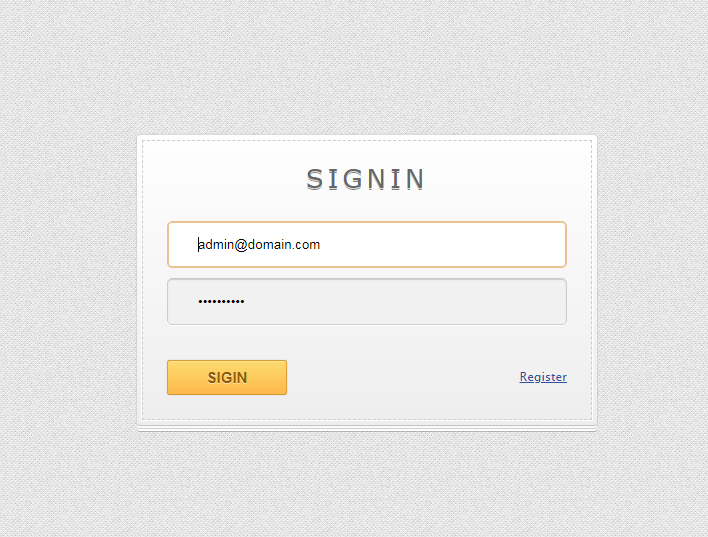
//[Authorize(Roles = "User;Admin")]

[HttpGet]

[AllowAnonymous]

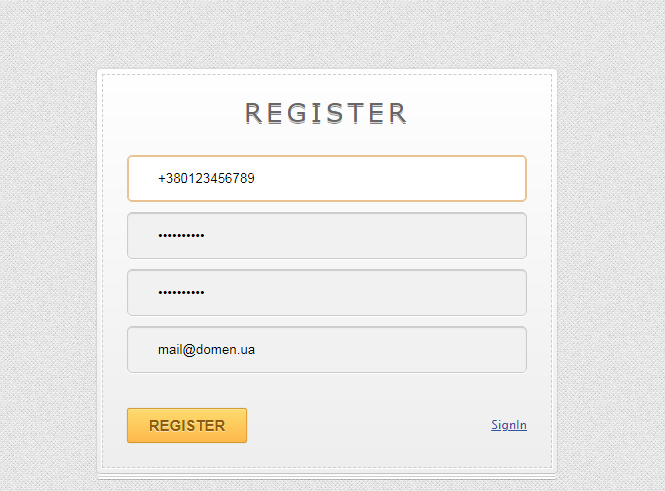
public async Task<IActionResult> CompletingTestAndGetResult(int generatedTestId)

**6. Руководство пользователя**

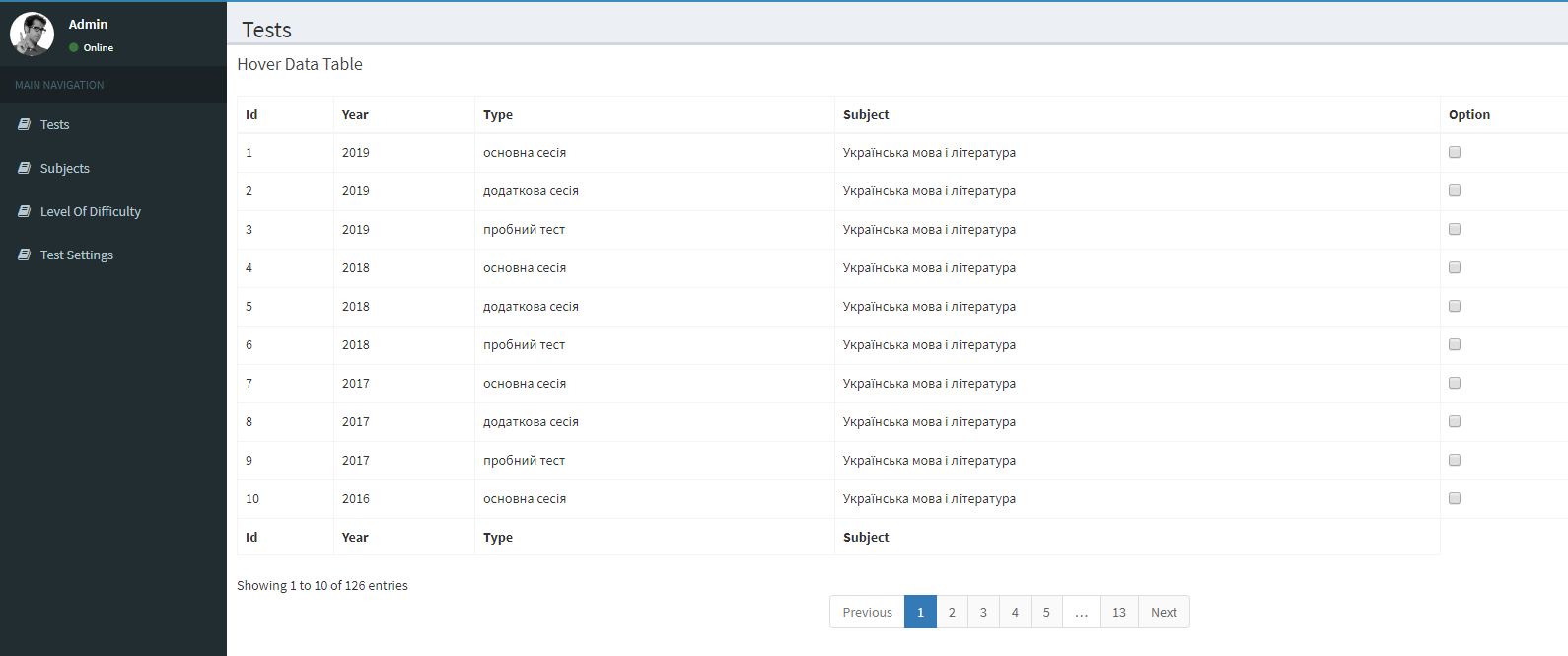


На главной странице сайта расположена формы для авторизации. Через нее заходят в систему и администраторы, и преподаватели, и пользователи.

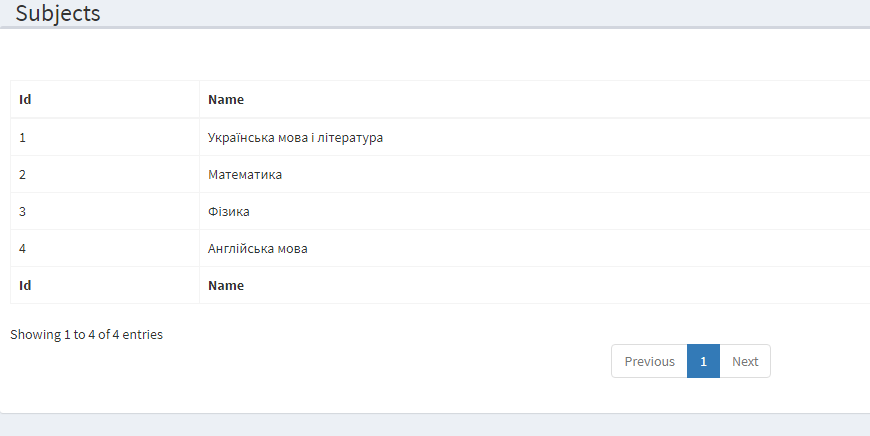
Если вы пользователь и вас еще нет в системе – нажмите на “Register” и вы попадете на следующую форму.



После удачной регистрации, автоматически произведется вход в систему. Если вы вошли под администратором вы увидите следующее:



Эта страница со всеми тестами.

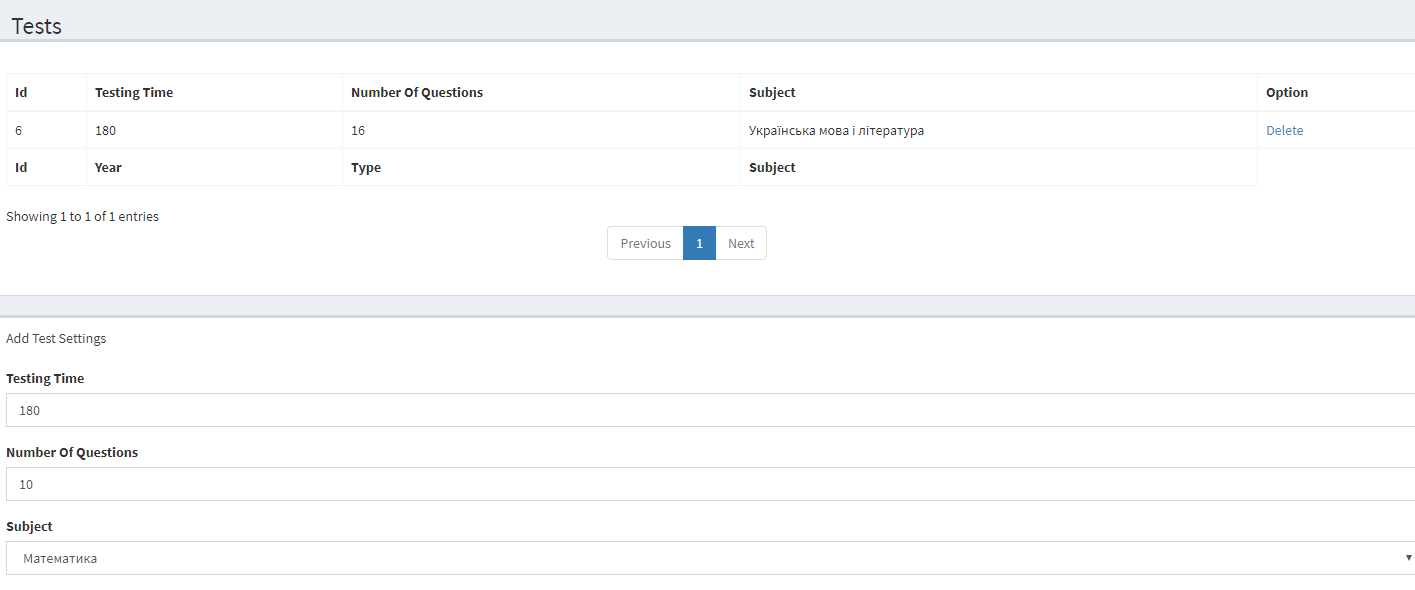


Предметы

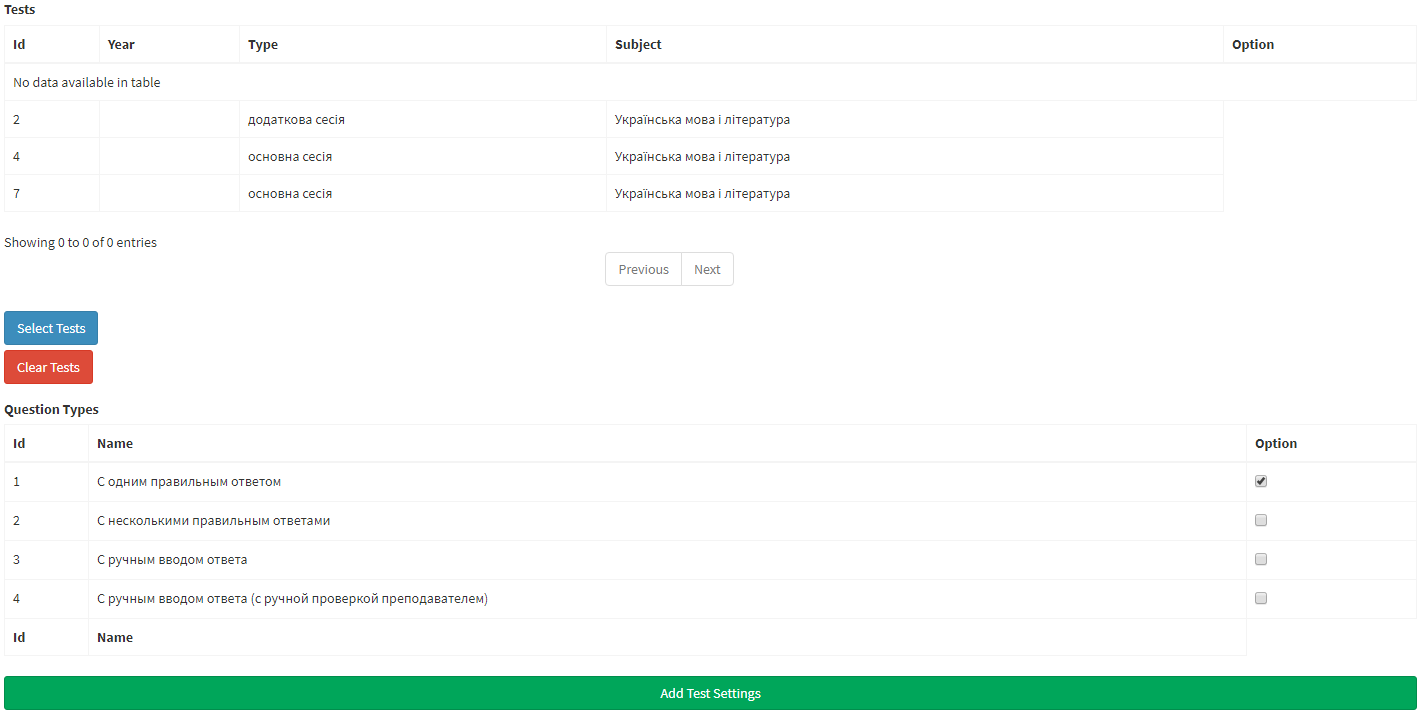


Типы вопросов

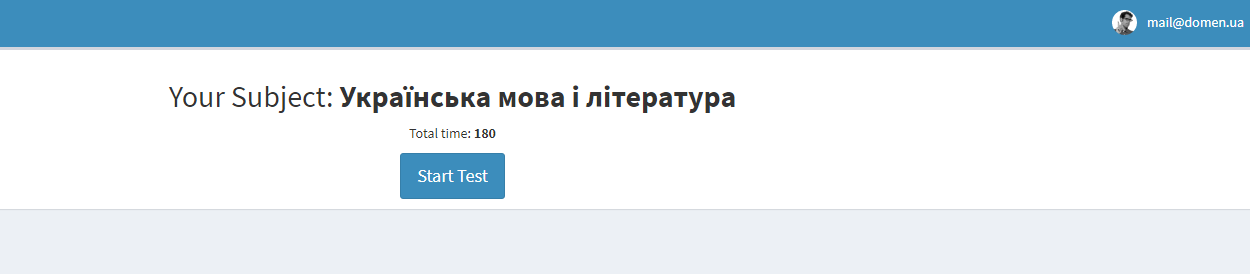
Перейдя во вкладу TestSettings – вы попадете в настройку теста. На данный момент можно добавить только для одного предмета и всего один тест. Чтобы внести изменения в текущий, предыдущий надо будет удалить.



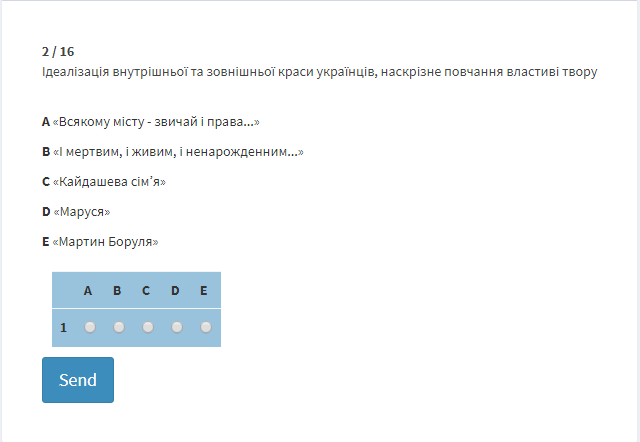
В форме ниже заполняются такие данные как время тестирования, количество вопросов в тесте, тесты и типы вопросов.



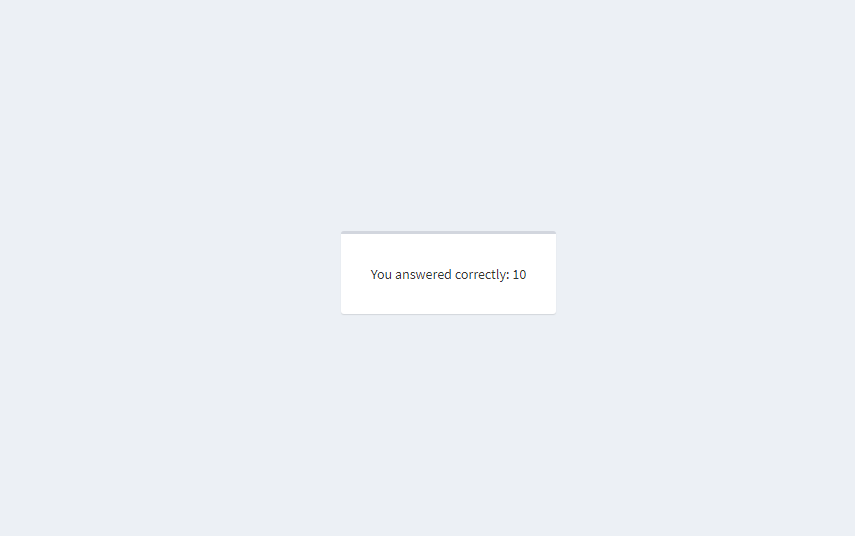
Если вы зашли как пользователь – у вас будет следующая страница



Если настройки теста не заданы – кнопка будет неактивна. Как только вы нажмете на кнопку – начнется тестирование.



После того как вы ответите на все вопросы – вам выдадут количество правильных ответов.



**7. Список литературы**

1) [www.msdn.com](http://www.msdn.com)

2) [www.metanit.com](http://www.metanit.com)

3) [www.professorweb.ru](http://www.professorweb.ru)

4) <https://habr.com/ru/>

5) <https://php.net/>

5) <http://www.php.su/>