

КУРСОВ ПРОЕКТ

ПО

Web технологии

преподавател: проф. д-р Милен Петров

СОФИЙСКИ
УНИВЕРСИТЕТ



„СВ. КЛИМЕНТ
ОХРИДСКИ“

ОСНОВАН 1888 Г.

Имена: Стефан Димитров Велев	имейл: sdvelev@uni-sofia.bg	ф.н.: 62537
Имена: Даниел Иванов Халачев	имейл: dihalachev@uni-sofia.bg	ф.н.: 62547
Начална година: 2023	Програма: бакалавър, (СИ)	Курс: 3
Тема: 3.2 KnowledgeTest	– Система за генериране на тестове	
Дата: 2023-06-23	Предмет: w20prj_SI_final	

Съдържание

1. Условие	3
2. Въведение	3
3. Теория	4
4. Използвани технологии	5
5. Инсталация и настройки.....	5
6. Кратко ръководство на потребителя	7
6.1 Вход и регистрация в системата	7
6.2 Решаване на тест в изпитен режим и възможност за обратна връзка	7
6.3 Възможност за преглед и ревюиране на отделни въпроси.....	9
6.4 Профилна страница.....	9
6.5 Страница с въпроси.....	10
6.6 Смяна на темата.....	11
7. Примерни данни	12
8. Описание на програмния код.....	13
9. Приноси на студента, ограничения и възможности за бъдещо разширение	21
10. Какво научих	22
11. Използвани източници	23

KnowledgeTest

1. Условие

Условието на проекта, така както е зададено от таблицата с теми за проекти, с леки модификации по реализацията, като възможност за интерпретация от страна на авторите, е:

Създаване на система за генериране на тестове - по зададен файл csv формат; и система за проиграване на тестове - в 'изпитен' режим и в режим 'рецензия' (възможност за обратна връзка на отговарящия - дали въпросът е коректен, каква е неговата сложност, възможност за репортуване на правописни и други грешки); Експорт на теста във формат за импорт/експорт от модул.

2. Въведение

В наши дни оценяването чрез тестове остава едно от основните средства за проверка на знанията в България. Със своите предимства и недостатъци, то остава много популярно във всякакви видове учебни заведения – от училища до университети. Процесът по организация на изпитните материали е малко или много строго индивидуален за всяка една организация.

За да се улесни и дигитализира работата с тестови въпроси и създаване на тестове, редица системи за генериране и решаване на изпити се разработват в днешно време. Такава е и нашата система **KnowledgeTest**. Акцентът на предложеното решение е начинът на импортиране на въпроси от CSV файл, както и целият процес по работа с тях.

Освен изпитният режим, който нашата система предлага, дава се възможност и за експортиране на въпроси към *Moodle XML* формат – един от поддържаните формати от обучителната платформа *Moodle*, който позволява наведнъж да бъдат вкарани множество от тестови въпроси. Това позволява на една такава система като нашата, да бъде посредник между въпроси, съхранявани във файл – най-простият вид база данни, и *Moodle*, една сложна и отличена за целта си платформа, която допълва процеса по полагане на тест.

Друг основен фокус на нашата система е и възможността за получаване на обратна връзка при подаден отговор. Това остава много важно и ценно, тъй като доста често коментарите след тест остават забравени от неговите полагачи. В никакъв случай не е най-ползното нещо да видиш само дали даденият отговор е верен или грешен, а е това да получиш последващи напътствия и коментари. Нашата платформа се придържа към

този принцип с възможността си да обработва тази обратна връзка и да я подава при експорт към *Moodle*.

3. Теория

Зад всеки един софтуерен продукт стои неговата софтуерна архитектура. За разработка на нашия проект **KnowledgeTest** предпочетохме архитектурния стил *REST* – *Representational State Transfer*. Причината за този наш избор е удобството, което придържането към основните принципи на тази архитектура би ни донесло.

Някои от основните принципи, които стоят зад този модел, са стандартизирани интерфейси, липса на запазване на състояние, многослоеста архитектура и др. Нека да разгледаме как биват приложени за нашия софтуер.

Стандартизираните интерфейси са фундаментална черта на т.нар. *RESTful* уеб услуги. Те са универсален начин за предаване на информация от базата данни под формата на ресурси. Това бива обобщено и документирано в т.нар. *endpoints* (крайни точки). В нашата система това е пътят, част от *URL-a* на страницата, */api*. С помощта на този програмен интерфейс, ние можем да подаваме заявки към сървъра от всякакви видове клиенти, като това не се ограничава само до нашия *front-end*.

Несъмнено всичко това поражда нуждата от подходяща защита, която се грижи за това да няма изтичане или обявяване на информация, която бива предназначена и е собственост на друг потребител. Това се осъществява чрез двете, използвани много често заедно, понятия: **автентикация** – установяване на самоличността на даден потребител, и **оторизация** – действията, които даден потребител може да извършва. В **KnowledgeTest** това се осъществява чрез използването на т.нар. *JSON Web Token* – *JSON-базиран отворен стандарт*.

Друг основен принцип на *REST* е липсата на поддръжка на сесии. Това означава, че всяка една заявка от сървъра би могла да се изпълни, независимо от останалите, без нейния успех да зависи от тях. Това придава известен смисъл на заявките като основна единица, която служи за предаване и работа с ресурси.

Слоестата архитектура означава, че се дава възможност на отговора от сървъра да бъде опакован и преобразуван от слой на слой, така че да пристигне под най-подходящата форма за клиента. Това спомага за *енкапсулацията* по отношение на скриване на операции и други действия в рамките на отделен слой. Такъв вид разделение и архитектура остава предпочитан вариант при разработката на голяма част от уеб приложенията в наши дни. Предложеното от нас решение се основава именно на тези добри подходи за работа. Прилагането им на практика означава за всеки един вид, категория ресурси, да се имплементира т.нар. *контролер*, който директно да комуникира с нашия клиент. Под него стои т.нар. *gateway* (шлюз), който в случая е директен посредник между контролера и базата данни. В него пряко бива заложена бизнес логиката, стояща зад нашето приложение. За контрол на дължината на отговора е използван принципа на т.нар. *pagination* (странициране).

Не на последно място стои и базата данни, която има отговорната задача да съхранява получената информация от заявките и да я предоставя при определени обстоятелства. Характерно за работа с нея при такъв тип архитектура е подаването и обмяната на *token-и* в заглавната част на всяка една заявка, която не е предназначена за всеки вид потребител. Такива обикновено са заявките по изтриване на ресурс, достъпване на ценна информация и др.

4. Използвани технологии

Проектът **KnowledgeTest** се опира на трислойната архитектура по отношение на начин на изграждане. За представителния слой са използвани *HTML*, *CSS* и *JavaScript*, като универсални технологии за всеки уеб клиент. С помощта на *JavaScript* се постига и комуникацията с приложния програмен интерфейс */api*.

Използваната технология за бизнес логиката е *PHP*, версия 8.2.0. В поддиректориите *libs/* и *pages/* бива представена тази част от уеб сайта.

За базата данни сме използвали *MySQL*, чрез който с поредица от скриптове, намиращи се в поддиректорията *db/*, построяваме схемата на нашата софтуерна система. Промени по нея могат да бъдат правени и чрез възможността за качване на файл в *CSV* формат.

5. Инсталация и настройки

Тъй като за разработката на приложението е използван *PHP 8* като технология, нужен е сървър за неговата поддръжка. За целта използваме *XAMPP v3.3.0* (възможно е и работа на предишна версия, стига тя да поддържа *PHP 8*) – уебсървърен пакет с отворен код, който включва *Apache/2.4.54 HTTP* сървър, база данни *MariaDB* (заменена по-късно от *MySQL*), технологиите *PHP* и *Perl*.

```
-- phpMyAdmin SQL Dump
-- version 5.2.0
-- https://www.phpmyadmin.net/
--
-- Host: 127.0.0.1
-- Generation Time: Jun 23, 2023 at 07:47 AM
-- Server version: 10.4.27-MariaDB
-- PHP Version: 8.2.0
```

След инсталацията на *XAMPP* е важно цялата директория на проекта **KnowledgeTest** да се добави в директорията *htdocs*, за да може след това софтуерното приложение да се стартира в браузъра по зададения път и локация.

Друга особеност на проекта е наличието в основната директория на файл, който служи за конфигуриране на модула за пренаписване на *URL* адреси на уеб сървъра *Apache*. Неговата роля е да включи механизма за пренаписване на *URL* и да зададе като базов адрес за всички останали правила директорията на проекта. Важно е да се спомене, че направеният от нас *.htaccess* файл може да не работи, ако в папката *htdocs* вече има друг *.htaccess* файл. Ако в действителност има конфликт, то препоръчваме файлът в *htdocs* да бъде временно преместен, или този в папката на нашия проект да бъде

редактиран, за да работи коректно заедно с другия. Следващото условие и последващо правило гласи, че ако предоставеният *URL* адрес започва с *KnowledgeTest/api*, то тогава потребителят се пренасочва към *api/index.php* директно, скрито от него. Там се обработват всички параметри от *URL-a* и се преминава към съответния контролер, в зависимост от значението им. Накрая на файла се настройва глобална променлива на средата *SECRET_KEY*, която след това може да бъде достъпвана чрез съвърнатата логика и която ще послужи в процеса по генериране на уникален *JWT Token*.

Важно: Поради особености на процеса по *URL Rewriting* в *.htaccess*, папката **KnowledgeTestImproved ТРЯБВА** да бъде поставена непосредствено в кореновата директория (**htdocs**), а не в друга вътрешна папка.

Конфигурациите се намират във файла: *libs/Settings.php*:

```
<?php
// SERVER CONFIGURATION VARIABLES
// Site Variables
define('SITE_NAME', 'KnowledgeTest');
define('SITE_CREATOR_NAME_1', 'Даниел Халачев');
define('SITE_CREATOR_FN_1', '62547');
define('SITE_CREATOR_EMAIL_1', 'dihalachev@uni-sofia.bg');
define('SITE_CREATOR_NAME_2', 'Стефан Велев');
define('SITE_CREATOR_FN_2', '62537');
define('SITE_CREATOR_EMAIL_2', 'sdvelev@uni-sofia.bg');
define('SITE_INFO', 'This project was created during 2023 year, on Web Technologies course at FMI, Sofia University, lead by: prof. Milen Petrov');
define('SITE_DESCRIPTION', 'Software system for generating, importing, exporting and solving tests');
// Database Variables
define('DB_HOST', 'localhost');
define('DB_NAME', 'tests');
define('DB_USER', 'root');
define('DB_PASSWORD', '');
// Default number of items in GET collection response
define('DEFAULT_QUERY_SIZE', 10);
// CSV Variables
define('DELIMITER', ';');
define('MAXIMUM_LINE_LENGTH', 10000);
define('FROM_ENCODING', 'CP866');
define('TO_ENCODING', 'UTF-8');
define('OPENING_MODE', 'r');
define('INDEX_COLUMN_TIME', 0);
define('INDEX_COLUMN_FACULTY_NUMBER', 1);
define('INDEX_COLUMN_QUESTION_NUMBER', 2);
define('INDEX_COLUMN_QUESTION_AIM', 3);
define('INDEX_COLUMN_QUESTION_TYPE', 4);
define('INDEX_COLUMN_QUESTION_TEXT', 5);
define('INDEX_COLUMN_QUESTION_FIRST_OPTION', 6);
define('INDEX_COLUMN_QUESTION_SECOND_OPTION', 7);
define('INDEX_COLUMN_QUESTION_THIRD_OPTION', 8);
define('INDEX_COLUMN_QUESTION_FOURTH_OPTION', 9);
define('INDEX_COLUMN_QUESTION_CORRECT_ANSWER', 10);
define('INDEX_COLUMN_QUESTION_COMPLEXITY', 11);
define('INDEX_COLUMN_CORRECT_FEEDBACK', 12);
define('INDEX_COLUMN_INCORRECT_FEEDBACK', 13);
define('INDEX_COLUMN_NOTES', 14);
?>
```

Предоставени са скриптове за създаване на базата данни и таблиците към нея (*createAllScript.sql*), както и скрипт с примерни данни (*insertDataScript.sql*) в директорията *db/* на проекта.

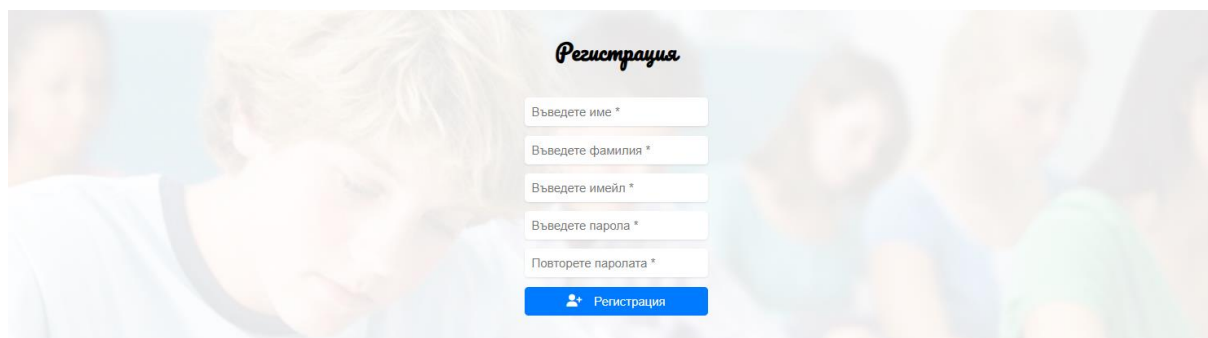
6. Кратко ръководство на потребителя

В следващите страници ще обобщим и групираме основните функционалности на софтуерния проект *KnowledgeTest*, обособени в отделни подсекции.

6.1 Вход и регистрация в системата

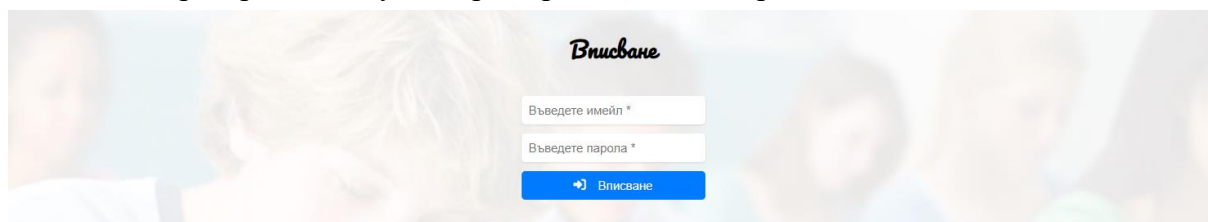
Разработената система за генериране на тестове предоставя възможност за регистрация и вписване в нея. Извършването на регистрация се налага при желание за създаване на тестове и управление на действията с тях. За решаване на тестове в изпитен режим и режим ревю не се изисква регистрация, тъй като системата предлага анонимност по отношение на тези действия. За доближаване до реална изпитна обстановка, може да се използва опцията за експорт към *Moodle XML* формат и там да бъде проведен реален изпит.

Формулярите за вход и регистрация са стандартните и общоприети за работа с тези действия. За регистрация в системата се изисква задължително въвеждането на име, фамилия, имейл, парола, както и тя да се потвърди чрез нейното повторно въвеждане.



Фигура 1: Формуляр за регистрация в *KnowledgeTest*

За формуляра за вписване е необходимо въвеждането на имейл и парола. С натискането на бутона за вписване, данните бива валидирани и проверени за съществуващ такъв запис в базата данни. Подходящи съобщения под формата на изскачащи прозорци се визуализират при всеки един проблем с данните.

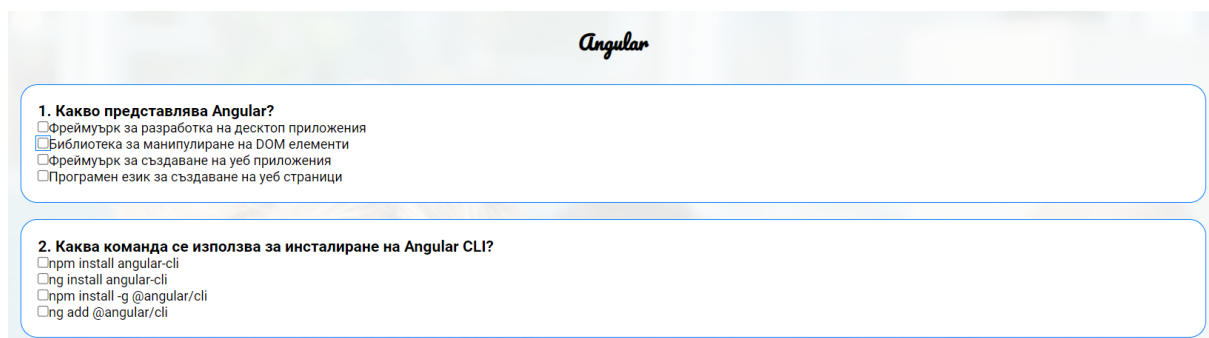


Фигура 2: Формуляр за вписване в *KnowledgeTest*

6.2 Решаване на тест в изпитен режим и възможност за обратна връзка

Решаването на тест в изпитен режим може да бъде достъпно от началната страница преди процесите по регистрация и вписване. В секцията за преглед на тест се изисква единствено въвеждането на код на теста. Той може да бъде получен от дадения преподавател след създаването на теста и качването на въпросите в него.

Изпитният режим е под формата на тест с отделни секции за визуализиране на въпрос, заедно с неговите отговори. Решаващият теста посочва верния според него отговор. При завършване на теста и изпращането му, той получава обратна връзка под формата на проценти успеваемост на теста. Няма ограничение в броя опити за решаване на теста, с цел затвърждаване на заложения обучителен материал. При посочване на грешен отговор на въпрос, верните отговори не се показват при предаване на теста, за да може да се насърчи повторното му решаване до постигането на максимален резултат.



Angular

1. Какво представлява Angular?

- ☐ Фреймвърк за разработка на десктоп приложения
- ☒ Библиотека за манипулиране на DOM елементи
- ☐ Фреймвърк за създаване на уеб приложения
- ☐ Програмен език за създаване на уеб страници

2. Каква команда се използва за инсталиране на Angular CLI?

- ☐ npm install angular-cli
- ☐ ng install angular-cli
- ☐ npm install -g @angular/cli
- ☐ ng add @angular/cli

Фигура 3: Решаване на тест в изпитен режим в KnowledgeTest

Режимът за рецензия на въпроси може да бъде достъпен за цял тест отново по същия начин, както изпитният режим, от началната страница чрез въвеждане на кода на теста и избор на бутона за ревю. За разлика от изпитния режим, тук верните отговори се визуализират. За да бъде направена обективна и коректна рецензия на въпрос, това е много важно. Чрез контролите за поставяне на сложност на въпрос от 1 до 10 и възможността за предоставяне на коментар в текстовото поле, за всеки въпрос се дава свобода на рецензиращия да опише недостатъците и положителните черти на въпроса. Там могат да бъдат посочени правописни и други видове грешки. Едно от най-важните неща е, че обратните връзки към въпроси са анонимни, което гарантира търсената обективност.



Angular

1. Какво представлява Angular?

- ☐ Фреймвърк за разработка на десктоп приложения
- ☐ Библиотека за манипулиране на DOM елементи
- ☒ Фреймвърк за създаване на уеб приложения
- ☐ Програмен език за създаване на уеб страници

Сложност от 1 до 10:

2. Каква команда се използва за инсталиране на Angular CLI?

- ☐ npm install angular-cli
- ☐ ng install angular-cli
- ☒ npm install -g @angular/cli
- ☐ ng add @angular/cli

Сложност от 1 до 10:

3. Какво представлява компонентът в Angular?

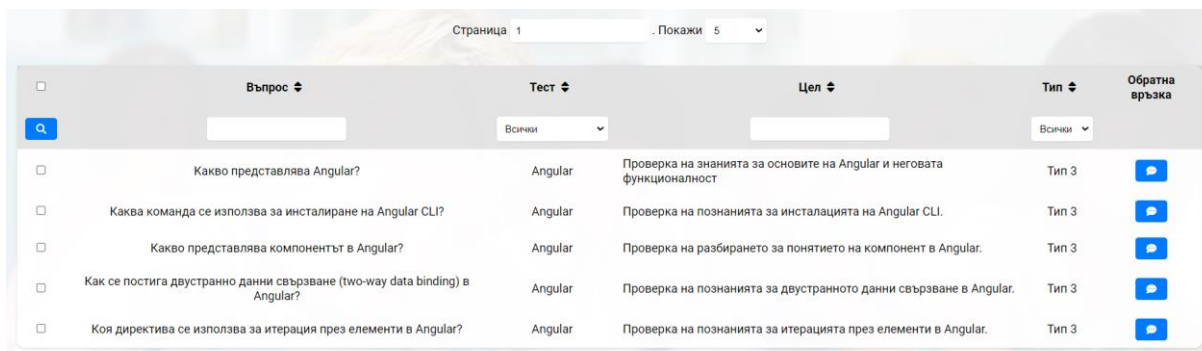
- ☐ Част от Angular модул
- ☐ Код за валидация на форми
- ☒ HTML шаблон за потребителски интерфейс
- ☐ Сървиз за извличане на данни от база данни

Сложност от 1 до 10:

Фигура 4: Рецензиране на тест в KnowledgeTest

6.3 Възможност за преглед и ревю на отделни въпроси

От началната страница има възможност за преглед на въпроси. Тази опция е независима от решаването и оценката на цял тест, тъй като тук се виждат всички въведени въпроси в системата. Целта е да може да се дава обратна връзка и да се преглеждат въпроси дори и потребителят да няма информация за код за започване на конкретен тест.



Страница 1 . Покази: 5

Въпрос	Тест	Цел	Тип	Обратна връзка
<input type="checkbox"/> Какво представлява Angular?	Angular	Проверка на знанията за основите на Angular и неговата функционалност	Тип 3	
<input type="checkbox"/> Каква команда се използва за инсталиране на Angular CLI?	Angular	Проверка на познанията за инсталацията на Angular CLI.	Тип 3	
<input type="checkbox"/> Какво представлява компонентът в Angular?	Angular	Проверка на разбирането за понятието на компонент в Angular.	Тип 3	
<input type="checkbox"/> Как се постига двустранно данни свързване (two-way data binding) в Angular?	Angular	Проверка на познанията за двустранното данни свързване в Angular.	Тип 3	
<input type="checkbox"/> Коя директива се използва за итерация през елементи в Angular?	Angular	Проверка на познанията за итерацията през елементи в Angular.	Тип 3	

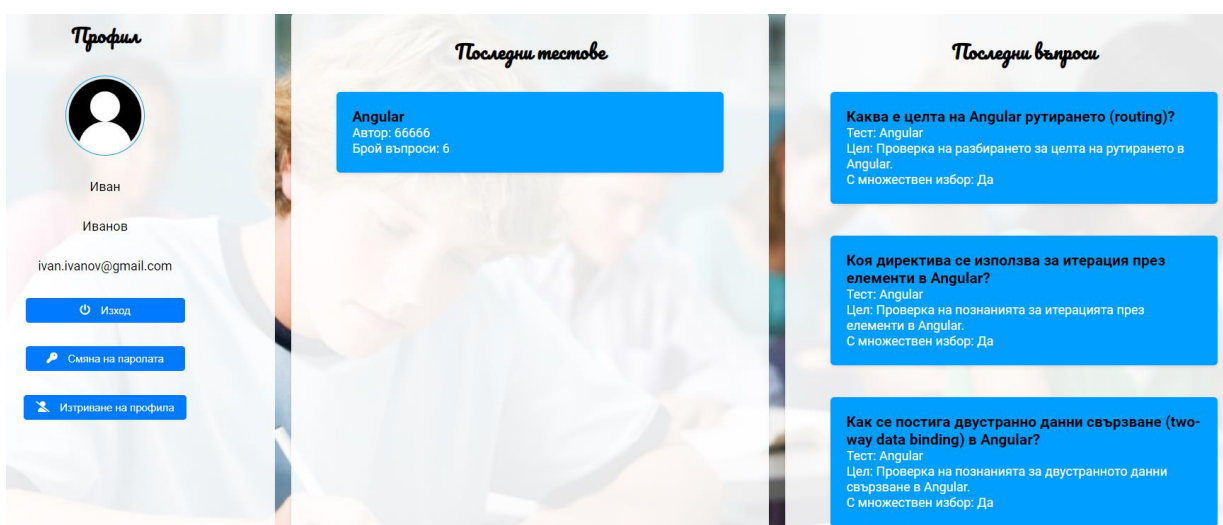
Фигура 5: Преглед на въпроси от началната страница на KnowledgeTest

6.4 Профилна страница

От профилната страница могат да се достъпят няколко функционалности. Една от тях е възможността за смяна на профилната снимка. Тази смяна може да се осъществи чрез натискане върху самата снимка и избиране на нова от файловата система.

Под нея се виждат детайлите за профила, а под тях стоят три бутона, съответно за изход, смяна на паролата и изтриване на профила. Смяната на паролата се осъществява чрез бутона за смяна и въвеждането на новата парола два поредни пъти. Изтриването на профила се случва чрез повторно потвърждение.

В секциите за последни тестове и въпроси може да бъде намерена накратко информацията за най-скорошната активност в рамките на профила.



Профил

Иван
Иванов
ivan.ivanov@gmail.com

Изход

Смяна на паролата

Изтриване на профила

Последни тестове

Angular
Автор: 66666
Брой въпроси: 6

Последни въпроси

Каква е целта на Angular рутването (routing)?
Тест: Angular
Цел: Проверка на разбирането за целта на рутването в Angular.
С множествен избор: Да

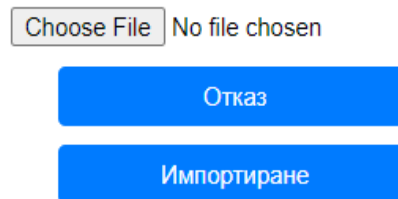
Коя директива се използва за итерация през елементи в Angular?
Тест: Angular
Цел: Проверка на познанията за итерацията през елементи в Angular.
С множествен избор: Да

Как се постига двустранно данни свързване (two-way data binding) в Angular?
Тест: Angular
Цел: Проверка на познанията за двустранното данни свързване в Angular.
С множествен избор: Да

Фигура 6: Профилна страница в KnowledgeTest

6.5 Страница с въпроси

От страницата с въпроси могат да бъдат извършени основните операции с тестовите и въпросите към тях. Могат да бъдат качвани въпроси от CSV файл чрез бутона *Качване*. Форматът и структурата на CSV файла е точно зададена и определена за проекта, но като бъдещо развитие може да бъде модифицирана, с цел добавянето на още полета, които по сегашния формат остават празни или със стойности по подразбиране.



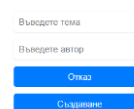
Фигура 7: Диалогов прозорец след избиране на "Качване" в KnowledgeTest

Структурата на CSV файла, който системата обработва, трябва да бъде следната:

Клеймо за време;Факултетен номер;Номер на въпроса;Цел на въпроса;Тип на въпроса;Въпрос;Опция 1;Опция 2;Опция 3;Опция 4;Верен отговор;Ниво на трудност;Обратна връзка при верен отговор;Обратна връзка при грешен отговор;Забележка

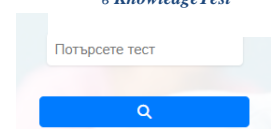
Като разделител се използва символът „;“. В конфигурациите на проекта всички данни, свързани с процеса по четенето, както и разположение на колони, са изнесени с цел по-лесното преструктуриране при промяна на формата. Примерен файл с тази структура може да бъде намерен в поддиректорията *csv/* на **KnowledgeTest**. Като резултат, качените тестове са с име във формат „YYYY-MM-DD-FN“, където датата е актуалната за деня, а факултетният номер е на автора на теста.

Друга функционалност е създаването на тест. При създаване на тест могат да бъдат посочени темата и автора на теста. След успешно създаване, в лявата част на екрана при другите тестове се виждат и новите метаданни.



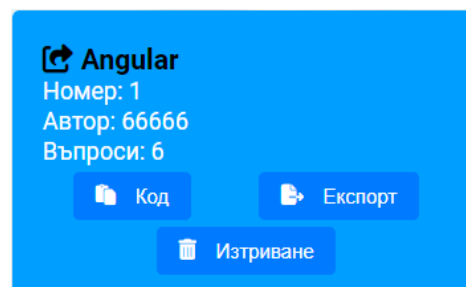
Фигура 8: Диалогов прозорец след избиране на "Нов тест" в KnowledgeTest

Възможността за търсене на тест се осъществява по име на темата на теста. Не е необходимо въвеждането на пълното име на темата, за да се визуализира въпросният тест. Това може да става само по част от името на темата.



Фигура 9: Поле и бутон за търсене на тест по тема в KnowledgeTest

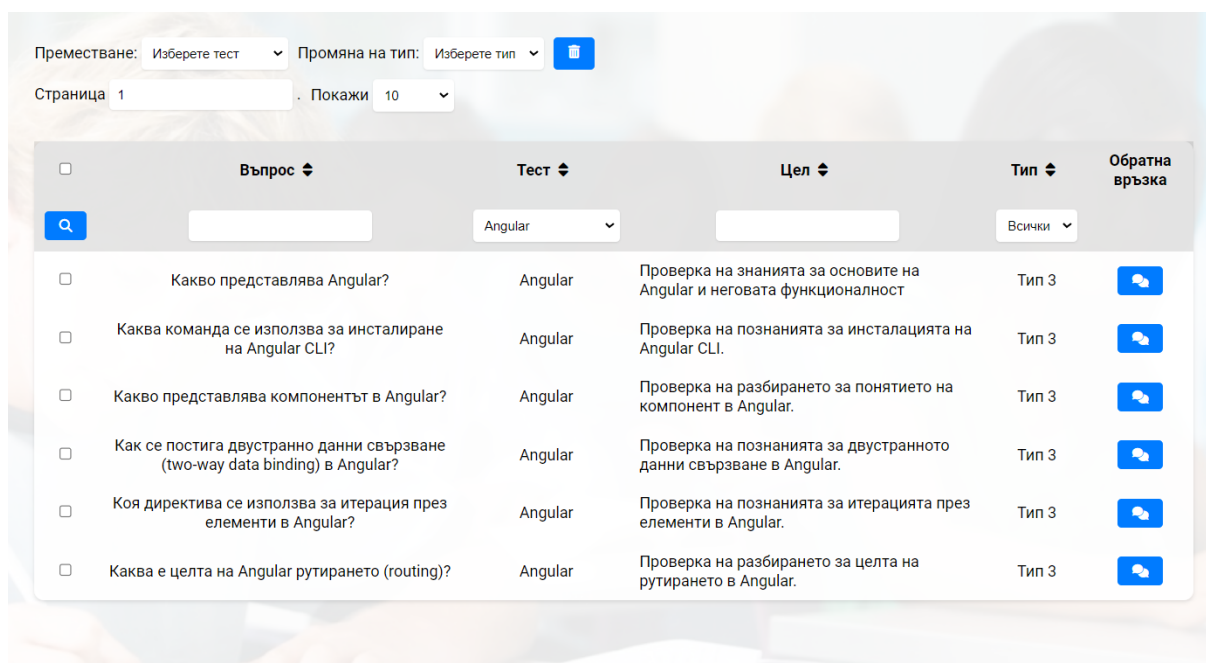
Обобщените данни за всеки тест се намират в лявата част на екрана. Към заглавието на всеки тест, което съвпада с темата, други метаданни са номера на теста, необходим за стартиране на теста в началната страница, авторът на теста, както и броят въпроси, които съдържа тестът. Бутоните за действие с теста са съответно *Код*, който позволява номерът на теста да се копира в буфера директно без да се налага маркиране, *Експорт*, който автоматично изтегля теста в *Moodle XML* формат, както и *Изтриване*, чрез който може да бъде изтрит тестът, заедно с всички въпроси и обратна връзка към него.



Фигура 10: Обобщени данни за тест в KnowledgeTest

В дясната част на екрана стоят въпросите, представени в табличен вид. При избор на даден въпрос могат да бъдат видени неговите отговори, заедно с верния отговор. Като заглавни елементи на първия ред на таблицата стоят контролите за търсене по съдържание на въпрос, филтриране по тест, към който принадлежи въпросът, цел и тип на въпроса. В последната колона стои бутонът за преглед на обратната връзка, подадена за съответния въпрос. Като първа колона на таблицата стоят бутони за отметка, откъдето могат да се маркират няколко въпроса наведнъж. След като бъдат маркирани, в горната част на страницата стоят операциите, които могат да бъдат извършени върху всички тях. Предоставя се възможност за преместване на въпрос от даден тест в друг, промяна на типа на въпроса и изтриване на въпроса.

Под тези контроли за операции с въпроси, стоят полетата, отговарящи за страницирането (*API pagination*). По подразбиране, на една страница се виждат по 10 въпроса. Това може да бъде променено от падащия списък, а номера на текущата страница – от текстовото поле от лявата му страна.



Преместване:	Изберете тест	Промяна на тип:	Изберете тип	
Страница	1	Покажи	10	

<input type="checkbox"/>	Въпрос	Тест	Цел	Тип	Обратна връзка
<input type="checkbox"/>	Какво представлява Angular?	Angular	Проверка на знанията за основите на Angular и неговата функционалност	Тип 3	
<input type="checkbox"/>	Каква команда се използва за инсталиране на Angular CLI?	Angular	Проверка на познанията за инсталацията на Angular CLI.	Тип 3	
<input type="checkbox"/>	Какво представлява компонентът в Angular?	Angular	Проверка на разбирането за понятието на компонент в Angular.	Тип 3	
<input type="checkbox"/>	Как се постига двустранно данни свързване (two-way data binding) в Angular?	Angular	Проверка на познанията за двустранното данни свързване в Angular.	Тип 3	
<input type="checkbox"/>	Коя директива се използва за итерация през елементи в Angular?	Angular	Проверка на познанията за итерацията през елементи в Angular.	Тип 3	
<input type="checkbox"/>	Каква е целта на Angular рутирането (routing)?	Angular	Проверка на разбирането за целта на рутирането в Angular.	Тип 3	

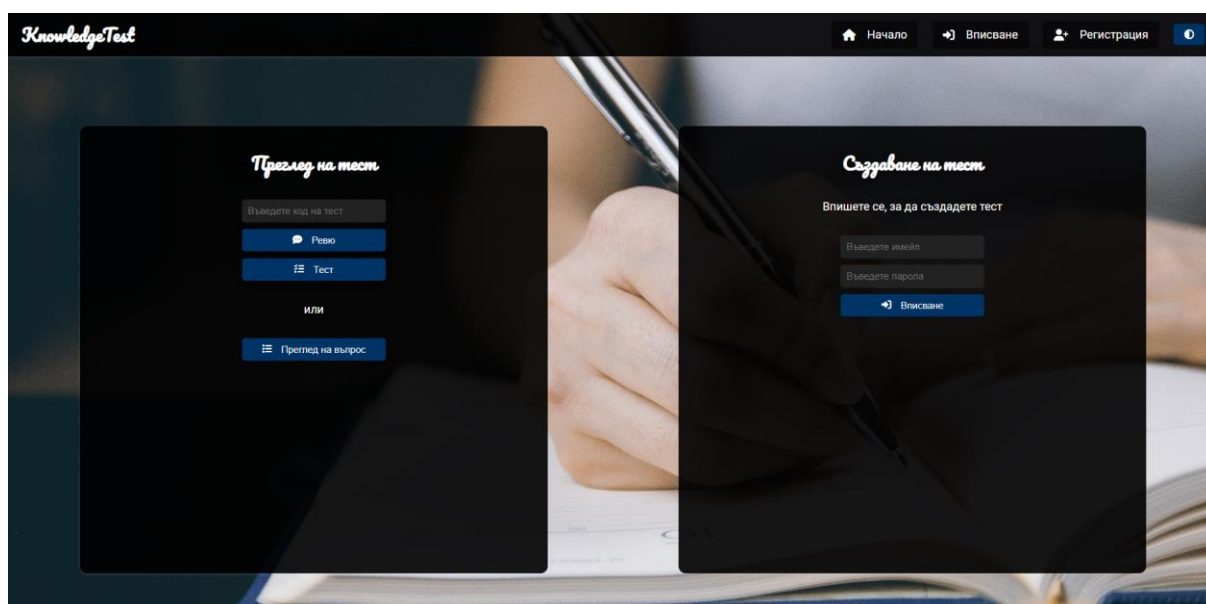
Фигура 11: Таблица с въпроси в страницата за въпроси на *KnowledgeTest*

6.6 Смяна на темата

Чрез бутон, намиращ се в горната дясна част на страницата, сайтът предоставя възможност за замяна на светлата тема с тъмна. Това се осъществява чрез бутон с икона, подсказваща функционалността му.



Фигура 12: Бутон за смяна на темата в *KnowledgeTest*



Фигура 13: Началната страница с тъмна тема в KnowledgeTest

7. Примерни данни

Примерни данни под формата на *SQL* скрипт са представени в поддиректорията *db/* – *insertDataScript.sql*. В този скрипт се създава потребител с име и фамилия Иван Иванов.

За вход в системата чрез този потребител се използват следните входни данни:

Имейл: ivan.ivanov@gmail.com

Парола: 123456

Освен потребител, който се записва в таблицата се създава и тест от негово име, който е с тема *Angular* и е качен от автор с факултетен номер 66666. Към него са добавени 6 въпроса, всеки от които има по един верен отговор.

Съдържанието на скрипта, съкратено до добавяне на първите три въпроса, е:

```
USE tests;

INSERT INTO `users` (id, email, password, firstName, lastName)
VALUES (1, "ivan.ivanov@gmail.com",
"$2y$10$3exSLh7UxY538n0soljHA0oloS8dGx6eX9vZ.O73U/PXRM2zS9OC.", "Иван", "Иванов");

INSERT INTO `tests` (id, uploaderId, author, topic)
VALUES (1, 1, "66666", "Angular");

INSERT INTO `questions` (id, testId, uploaderId, aim, questionType,
isMultipleChoice, label, correctFeedback, incorrectFeedback)
VALUES (1, 1, 1, "Проверка на знанията за основите на Angular и неговата
функционалност", 3, 1, "Какво представлява Angular?",
"Правилно! Angular е фреймуърк за създаване на уеб приложения.", "Трешка! Angular е
фреймуърк за създаване на уеб приложения.");

INSERT INTO `answers` (id, questionId, uploaderId, label, isCorrect)
VALUES (1, 1, 1, "Фреймуърк за разработка на десктоп приложения", 0);
```

```

INSERT INTO `answers` (id, questionId, uploaderId, label, isCorrect)
VALUES (2, 1, 1, "Библиотека за манипулиране на DOM елементи", 0);

INSERT INTO `answers` (id, questionId, uploaderId, label, isCorrect)
VALUES (3, 1, 1, "Фреймуърк за създаване на уеб приложения", 1);

INSERT INTO `answers` (id, questionId, uploaderId, label, isCorrect)
VALUES (4, 1, 1, "Програмен език за създаване на уеб страници", 0);

INSERT INTO `feedbacks` (id, questionId, complexity, feedback)
VALUES (1, 1, 6, "Well-written!");

INSERT INTO `questions` (id, testId, uploaderId, aim, questionType,
isMultipleChoice, label, correctFeedback, incorrectFeedback)
VALUES (2, 1, 1, "Проверка на познанията за инсталацията на Angular CLI.", 3, 1,
"Каква команда се използва за инсталиране на Angular CLI?",
"Правилно! За инсталация на Angular CLI се използва командата npm install -g
@angular/cli", "Грешка! За инсталация на Angular CLI трябва да използвате командата
npm install -g @angular/cli.");

INSERT INTO `answers` (id, questionId, uploaderId, label, isCorrect)
VALUES (5, 2, 1, "npm install angular-cli", 0);

INSERT INTO `answers` (id, questionId, uploaderId, label, isCorrect)
VALUES (6, 2, 1, "ng install angular-cli", 0);

INSERT INTO `answers` (id, questionId, uploaderId, label, isCorrect)
VALUES (7, 2, 1, "npm install -g @angular/cli", 1);

INSERT INTO `answers` (id, questionId, uploaderId, label, isCorrect)
VALUES (8, 2, 1, "ng add @angular/cli", 0);

INSERT INTO `feedbacks` (id, questionId, complexity, feedback)
VALUES (2, 2, 7, "Excellent!");

INSERT INTO `questions` (id, testId, uploaderId, aim, questionType,
isMultipleChoice, label, correctFeedback, incorrectFeedback)
VALUES (3, 1, 1, "Проверка на разбирането за понятието на компонент в Angular.", 3,
1, "Какво представлява компонентът в Angular?",
"Правилно! Компонентът в Angular представлява HTML шаблон за потребителски
интерфейс.", "Грешка! В компонента в Angular е включен HTML шаблон за потребителски
интерфейс.");

INSERT INTO `answers` (id, questionId, uploaderId, label, isCorrect)
VALUES (9, 3, 1, "Част от Angular модул", 0);

INSERT INTO `answers` (id, questionId, uploaderId, label, isCorrect)
VALUES (10, 3, 1, "Код за валидация на форми", 0);
INSERT INTO `answers` (id, questionId, uploaderId, label, isCorrect)
VALUES (11, 3, 1, "HTML шаблон за потребителски интерфейс", 1);

INSERT INTO `answers` (id, questionId, uploaderId, label, isCorrect)
VALUES (12, 3, 1, "Сървиз за извличане на данни от база данни", 0);

INSERT INTO `feedbacks` (id, questionId, complexity, feedback)
VALUES (3, 3, 8, "Excellent but kind of misleading!");

```

8. Описание на програмния код

Софтуерният проект *KnowledgeTest* се състои от няколко поддиректории. Основната сървърна логика на *PHP* се намира в папката *libs/*. Както споменахме по-рано, следваният архитектурен стил е *REST* и като такъв, обуславя наличието на контролери и класове с основната бизнес логика на приложението. Именно затова, тук се намират класове като *AnswerController*, *FeedbackController*, *QuestionController*,

QuestionTypeController, *TestController*, *UserController* и съответните класове услуги *AnswerGateway*, *FeedbackGateway*, *QuestionGateway*, *QuestionTypeGateway*, *TestGateway*, *UserGateway*.

В съдържанието на един такъв контролер се обработват постъпили заявки към дефинираното *api/*. Те могат да бъдат **GET**, **POST**, **PATCH** и **DELETE**. Чрез наследяване на класа *ControllerBase* се избягва дублирането на логика, характерна за всеки един контролер – защита на данните чрез оторизация. По този начин гарантираме, че не може всеки, който направи заявка към даден адрес, да може да изтрие потребител, тест, въпрос и т.н. Това се случва чрез метода *verifyPayload()* на класа *ControllerBase*:

```
protected function verifyPayload(?array $payload, ?int $resourceOwnerId = null):
bool {
    if (!$payload) {
        $this->sendUnauthorizedResponse();
        return false;
    }
    if (!$payload["expiration"] || strtotime($payload["expiration"]) > time()) {
        $this->sendUnauthorizedResponse();
        return false;
    }
    if ($resourceOwnerId && $payload["userId"] && $payload["userId"] !==
$resresourceOwnerId) {
        $this->sendForbiddenResponse();
        return false;
    }
    return true;
}
```

Целта на класовете, съдържащи нашата бизнес логика, е да съставят и изпълняват заявки към базата данни. След тяхното изпълнение, едно ниво по-надолу, вече не се прави верификация на потребител, тъй като тя вече е била направена в контролера. При такъв вид класове основните методи са свързани с ролята на методите на основните *HTTP* методи. Имплементацията на заявките позволява подаването на множество параметри на заявката в адреса:

```
public function get(int $id): array | false {
    $sql = "SELECT * FROM `answers` WHERE id = ?";
    $statement = $this->connection->prepare($sql);
    $statement->bindParam(1, $id, PDO::PARAM_INT);
    $statement->execute();
    $data = $statement->fetch();
    if ($data) {
        $data["isCorrect"] = (bool) $data["isCorrect"];
    }
    return $data;
}

public function create(array $data): int {
    $sql = "INSERT INTO `answers` (uploaderId, questionId, label, isCorrect) VALUES
(:uploaderId, :questionId, :label, :isCorrect)";
    $statement = $this->connection->prepare($sql);
    $statement->bindParam(":uploaderId", $data["uploaderId"], PDO::PARAM_INT);
    $statement->bindParam(":questionId", $data["questionId"], PDO::PARAM_INT);
    $statement->bindParam(":label", $data["label"], PDO::PARAM_STR);
    $statement->bindParam(":isCorrect", $data["isCorrect"], PDO::PARAM_BOOL);
    $statement->execute();
    return $this->connection->lastInsertId();
}

public function update(array $current, array $new): int {
```



```

    $sql = "UPDATE `answers` SET uploaderId = :uploaderId, questionId = :questionId,
    label = :label, isCorrect = :isCorrect WHERE id = :id";
    $statement = $this->connection->prepare($sql);
    $statement->bindValue(":uploaderId", $new["uploaderId"] ??
$current["uploaderId"], PDO::PARAM_INT);
    $statement->bindValue(":questionId", $new["questionId"] ??
$current["questionId"], PDO::PARAM_INT);
    $statement->bindValue(":label", $new["label"] ?? $current["label"],
PDO::PARAM_STR);
    $statement->bindValue(":isCorrect", $new["isCorrect"] ?? $current["isCorrect"],
PDO::PARAM_BOOL);
    $statement->bindValue(":id", $current["id"], PDO::PARAM_INT);
    $statement->execute();
    return $statement->rowCount();
}

public function delete(int $id): int {
    $sql = "DELETE FROM `answers` WHERE id = :id";
    $statement = $this->connection->prepare($sql);
    $statement->bindValue(":id", $id, PDO::PARAM_INT);
    $statement->execute();
    return $statement->rowCount();
}

```

Чрез класа *JWT* моделираме един от утвърдените начини за автентикация и оторизация в уеб технологиите. Структурата на т.нар. *token* е съставена най-общо от три части – *header*, *payload* и *signature*. Заглавната част се състои от името на използвания алгоритъм и типа *token*. В тялото се съхраняват данни за *id-to* на вписания потребител, имейл адреса му, както и времето на изтичане на *token-a*. В нашия случай това е три дни след неговото успешно създаване. Дотогава сесията може да бъде запазена, ако не се осъществи принудителен изход от профила.

```

public static function encode($payload, $secret, $algorithm = 'HS256')
{
    $header = [
        'alg' => $algorithm,
        'typ' => 'JWT'
    ];
    $encodedHeader = self::base64UrlEncode(json_encode($header));
    $encodedPayload = self::base64UrlEncode(json_encode($payload));
    $signature = self::generateSignature($encodedHeader, $encodedPayload, $secret,
$algorithm);

    return $encodedHeader . '.' . $encodedPayload . '.' . $signature;
}

```

Класовете *Test* и *User* играят ролята на модели за по-важните ресурси в системата. Във файла *Settings.php* са дефинирани именуванни константи, характерни за софтуера. В *readCSV.php* и *CSVUploader.php* се осъществява възможността за качване на CSV файл с въпроси по зададен формат. За основните операции по извличане на данните се разчита на вградени функции като *fgetcsv()*, която позволява четене на отделните колони на CSV файла. Чрез класа *DatabaseHandler* се осъществяват *SQL* заявки, необходими за обновяване на съдържанието на базата данни.

```

private static function readCSV(string $fileName, int $uploaderId) : bool {
    if (($handle = fopen($fileName, OPENING_MODE)) !== FALSE) {
        $row = 0;
        while (($data = fgetcsv($handle, MAXIMUM_LINE_LENGTH, DELIMITER)) !==
FALSE) {
            $numberOfFields = count($data);
            $row++;
            if ($row === 1) {

```

```

        continue;
    }
    for ($c = 0; $c < $numberOfFields; $c++) {
        $data[$c] = mb_convert_encoding($data[$c], TO_ENCODING,
FROM_ENCODING);
    }
    $testId = DatabaseHandler::getTestId($uploaderId,
$data[INDEX_COLUMN_FACULTY_NUMBER]);
    $questionTypeId =
DatabaseHandler::getQuestionTypeId($data[INDEX_COLUMN_QUESTION_TYPE]);
    $questionId = DatabaseHandler::createQuestion(
        $testId,
        $uploaderId,
        $data[INDEX_COLUMN_QUESTION_AIM],
        $questionTypeId,
        true,
        $data[INDEX_COLUMN_QUESTION_TEXT],
        $data[INDEX_COLUMN_CORRECT_FEEDBACK],
        $data[INDEX_COLUMN_INCORRECT_FEEDBACK]
    );
    for ($i = 0; $i < 4; $i++) {
        $isCorrect = false;
        if ($i === $data[INDEX_COLUMN_QUESTION_CORRECT_ANSWER] - 1) {
            $isCorrect = true;
        }
        DatabaseHandler::createAnswer($questionId, $uploaderId
, $data[INDEX_COLUMN_QUESTION_FIRST_OPTION + $i],
            $isCorrect);
    }
    DatabaseHandler::createFeedback($questionId,
$data[INDEX_COLUMN_QUESTION_COMPLEXITY]);
    }
    fclose($handle);
    return true;
}
return false;
}

```

В директорията *pages/* се намира визуалната част на уеб сайта. В страницата *index.php* се намира началната страница. За вмъкване на задължителни елементи като *header* и *footer* се използва съврърна логика на *PHP*, докато за изпращане на заявки и за придаване на динамика се използва *JavaScript*. Характерна особеност е изпращането на асинхронни заявки чрез *JavaScript* с функцията *fetch()*:

```

function deleteQuestions (idList) {
    idList.forEach(function(id) {
        var url = '../api/questions/' + parseInt(id);
        fetch(url, {
            method: 'DELETE',
            headers: {
                Authorization: `Bearer ${getToken()}`
            }
        })
        .then(function(response) {
            if (response.ok) {
                location.reload();
            } else {
                console.error('Failed to delete question ' + id + '.');
            }
        })
        .catch(function(error) {
            console.error('An error occurred while deleting question ' + id + ':',
error);
        });
    });
}

```


В директорията *css/* е приложено цялото стилизиране на уеб приложението в отделни файлове, сочещи към съответната страница от *pages/*, като цветовете са зададени в *css/common.css* чрез CSS променливи:

```
body {
  display: grid;
  grid-template-rows: auto 1fr auto;
  grid-template-areas:
    "header"
    "main"
    "footer";
}

header {
  grid-template: header;
}

main {
  grid-template: main;
}

footer {
  grid-template: footer;
}

main {
  display: grid;
  grid-template-rows: 1fr;
  grid-template-areas:
    "content";
  padding: 20px 20px;
  gap: 20px;
}

@media (min-width: 768px) {
  main {
    grid-template-rows: 1fr;
    grid-template-columns: 1fr;
    grid-template-areas:
      "content";
  }
  section.activity {
    height: 80%;
    width: 80%;
    margin: auto;
  }
}
```

В директорията *db/* се намират скриптовете за началните операции с базата данни:

```
CREATE DATABASE tests;

USE tests;

CREATE TABLE users (
  id INT NOT NULL AUTO_INCREMENT,
  googleId VARCHAR(255) UNIQUE,
  email VARCHAR(255) UNIQUE NOT NULL,
  password VARBINARY(255) NOT NULL,
  firstName VARCHAR(50) NOT NULL,
  lastName VARCHAR(50) NOT NULL,
  profilePicture VARCHAR(25) NULL,
  PRIMARY KEY(id)
);

CREATE TABLE tests (
```

```

        id INT NOT NULL AUTO_INCREMENT,
        uploaderId INT NOT NULL,
        author VARCHAR(100) NULL,
        topic VARCHAR(50) NULL,
        PRIMARY KEY (id),
        FOREIGN KEY (uploaderId) REFERENCES users (id)
    );

...

```

В директорията *img/* се намират снимкови ресурси, докато в директорията *csv/* се намира примерен CSV файл в задължителната за системата структура.

В директорията *api/* се намира основната логика, пренасочваща към работа с приложния програмен интерфейс:

```

<?php

declare(strict_types=1);

spl_autoload_register(function ($class) {
    require_once "../libs/$class.php";
});

set_error_handler("ErrorHandler::handleError");
set_exception_handler("ErrorHandler::handleException");

header("Content-type: application/json; charset=UTF-8");

$parts = explode("/", $_SERVER["REQUEST_URI"]);
$id = $parts[4] ?? null;

$databaseConnection = new DatabaseConnection();

$userGateway = new UserGateway($databaseConnection);
$answerGateway = new AnswerGateway($databaseConnection);
$questionGateway = new QuestionGateway($databaseConnection);
$testGateway = new TestGateway($databaseConnection);
$feedbackGateway = new FeedbackGateway($databaseConnection);
$questionTypeGateway = new QuestionTypeGateway($databaseConnection);

session_start();
switch (explode("?", $parts[3])[0]) {
    case "users":
        $controller = new UserController($userGateway);
        $controller->processRequest($_SERVER["REQUEST_METHOD"], $_GET, $id);
        break;
    case "answers" :
        $controller = new AnswerController($answerGateway);
        $controller->processRequest($_SERVER["REQUEST_METHOD"], $_GET, $id);
        break;
    case "questions":
        $controller = new QuestionController($questionGateway);
        $controller->processRequest($_SERVER["REQUEST_METHOD"], $_GET, $id);
        break;
    case "tests":
        $controller = new TestController($testGateway);
        $controller->processRequest($_SERVER["REQUEST_METHOD"], $_GET, $id);
        break;
    case "feedback":
        $controller = new FeedbackController($feedbackGateway);
        $controller->processRequest($_SERVER["REQUEST_METHOD"], $_GET, $id);
        break;
    case "types":
        $controller = new QuestionTypeController($questionTypeGateway);
        $controller->processRequest($_SERVER["REQUEST_METHOD"], $_GET, $id);

```

```

        break;

default:
    http_response_code(404);
    exit;
    break;
}

```

В директорията *includes/* се намират двата вида горни колонтитули – за вписан и невписан потребител, както и долният колонтитул:

```

<?php
include_once "../libs/Settings.php";
?>
<footer>
    <p><?php echo SITE_NAME?> 2023&copy;</p>
    <p>
        <span><?php echo SITE_CREATOR_NAME_1?>, </span>
        <span><?php echo SITE_CREATOR_NAME_2?></span>
    </p>
</footer>

```

В директорията *js/* се намират необходимите скриптове, управляващи работата по *front-end-a*:

```

function getToken() {
    return document.cookie
        .split("; ")
        .find((row) => row.startsWith("token="))
        .split("=")[1];
}

function getQuestions(testId) {
    return fetch(`../api/questions?testId=${testId}&size=-1`, {
        headers: {
            Authorization: `Bearer ${getToken()}`,
        },
    })
        .then(response => response.json())
        .then(data => data)
        .catch(error => {
            console.error('Error fetching questions:', error);
            return [];
        });
}

function getAnswers(questionId) {
    return fetch(`../api/answers?questionId=${questionId}`, {
        headers: {
            Authorization: `Bearer ${getToken()}`,
        },
    })
        .then(response => response.json())
        .then(data => data)
        .catch(error => {
            console.error('Error:', error);
            return [];
        });
}

function generateMoodleXML(questions, topic) {
    const xml = document.createElement('quiz');
    xml.appendChild(document.createTextNode('\n'));
    const questionTopicElement = document.createElement('question');
    questionTopicElement.appendChild(document.createTextNode('\n'));
    questionTopicElement.setAttribute('type', 'category');
    xml.appendChild(questionTopicElement);
}

```

```

const questionCategoryElement = document.createElement('category');
questionCategoryElement.appendChild(document.createTextNode('\n'));
const textElement = document.createElement('text');
textElement.appendChild(document.createTextNode('\n'));
textElement.textContent = topic;
questionCategoryElement.appendChild(textElement);
questionTopicElement.appendChild(questionCategoryElement);
xml.appendChild(questionTopicElement);
xml.appendChild(document.createTextNode('\n'));

questions.forEach(question => {
  const questionElement = document.createElement('question');
  questionElement.appendChild(document.createTextNode('\n'));
  questionElement.setAttribute('type', 'multichoice');
  xml.appendChild(questionElement);

  const name = document.createElement('name');
  name.appendChild(document.createTextNode('\n'));
  const text = document.createElement('text');
  text.appendChild(document.createTextNode('\n'));
  text.textContent = question.label;
  name.appendChild(text);
  questionElement.appendChild(name);
  questionElement.appendChild(document.createTextNode('\n'));

  const questionTextElement = document.createElement('questiontext');
  questionTextElement.appendChild(document.createTextNode('\n'));
  questionTextElement.setAttribute('format', 'plain_text');
  const questionText = document.createElement('text');
  questionText.appendChild(document.createTextNode('\n'));
  questionText.textContent = question.label;
  questionTextElement.appendChild(questionText);
  questionElement.appendChild(questionTextElement);
  questionElement.appendChild(document.createTextNode('\n'));

  const answerNumberingElement = document.createElement('answernumbering');
  answerNumberingElement.appendChild(document.createTextNode('\n'));
  answerNumberingElement.textContent = 'ABCD';
  questionElement.appendChild(answerNumberingElement);
  questionElement.appendChild(document.createTextNode('\n'));

  const numberOfCorrectAnswers = question.answers.filter(answer =>
answer.isCorrect === true).length;

  question.answers.forEach(answer => {
    const answerElement = document.createElement('answer');
    answerElement.appendChild(document.createTextNode('\n'));
    answerElement.setAttribute('fraction', answer.isCorrect ? 100.0 /
numberOfCorrectAnswers : 0);
    answerElement.setAttribute('format', 'plain_text');
    const answerText = document.createElement('text');
    answerText.appendChild(document.createTextNode('\n'));
    answerText.textContent = answer.label;
    answerElement.appendChild(answerText);

    const feedback = document.createElement('feedback');
    feedback.appendChild(document.createTextNode('\n'));
    const feedbackText = document.createElement('text');
    feedbackText.appendChild(document.createTextNode('\n'));
    feedbackText.textContent = answer.isCorrect ? question.correctFeedback
: question.incorrectFeedback;
    feedback.appendChild(feedbackText);
    answerElement.appendChild(feedback);
    questionElement.appendChild(answerElement);
    questionElement.appendChild(document.createTextNode('\n'));
  });
});

```

```

        xml.appendChild(questionElement);
        xml.appendChild(document.createTextNode('\n'));
    });

    const serializer = new XMLSerializer();
    const xmlDeclaration = '<?xml version="1.0" encoding="UTF-8"?>\n';
    return xmlDeclaration + serializer.serializeToString(xml);
}

function exportToMoodleXML(testId, topic) {
    getQuestions(testId)
        .then(questions => {
            if (questions.length === 0) {
                console.log('No questions found.');
```

Извън тези директории се намира файлът *.htaccess*, за който вече споменахме по-рано. Той служи за необходимия процес по *URL Rewriting*:

```

RewriteEngine On
RewriteBase /KnowledgeTestImproved/

# Redirect requests to index.php
RewriteCond %{REQUEST_URI} ^/KnowledgeTestImproved/api [NC]
RewriteRule ^api(.*)$ api/index.php [L]
SetEnv SECRET_KEY "5acz+1HjfPR+y/JY2F+5odjl2P81jNInDQr4WGSxWlQ="
```

9. Приноси на студента, ограничения и възможности за бъдещо разширение

Разработката на проекта включваше разнообразни дейности и работа с различни технологии, които, обединени заедно, формират един цялостен софтуерен продукт, който може да бъде внедрен и поддържан. Решението за *REST* архитектура взехме поради изискването за скалируемост и възможност за интегриране с други системи. Относно имплементацията, ние се вдъхновихме от примерна разработка и обяснение в *YouTube* на Dave Hollingworth, озаглавено *Create a PHP REST API*. Наша беше задачата и впоследствие приносът да приложим тази техника на практика за нашия софтуер и да я надградим с възможности за сортиране, филтриране и странициране на заявките.

Избраният начин за автентикация и оторизация чрез *JWT Token* беше нещо ново за нас. За първи път реализирахме такава имплементация, която се базира на *token*. Намирането на подходящи и достатъчно детайлно описани инструкции по тази тема беше предизвикателство.

Като едно от ограниченията, което се превърна и в положителен елемент, беше избягването на употребата и зависимостта от външни библиотеки. Това се появи на дневен ред като въпрос в моментите, когато от *PHP* искахме да изпратим заявка към нашето */api* отново написано на *PHP*. За да се направи това, трябваше да се използват готови инструменти като *Curl* и др. решения, които предпочетохме да не използваме, именно със споменатата цел. Вместо това, навсякъде използвахме *fetch* заявки, имплементирани на *JavaScript*, като универсален скриптов език, който се разбира от всеки уеб клиент.

Като възможност за бъдещо развитие може да се работи в няколко посоки. От една страна, може да се подобри потребителският интерфейс и преживяване. От друга страна, може да се внедрят нови функционалности в интерфейса, например като възможност за създаване на въпроси (и в настоящия вариант това е възможно чрез изпращането на *POST* заявка към *api-to*). Друга възможна посока за работа е внедряването на изцяло нови услуги, като авторизация на всички потребители, запазване на резултати от тест и др.

10. Какво научих

По време на работата по проекта се изискваше интензивна и задълбочена работа с няколко различни технологии. Научихме някои нови неща за езика *PHP*, за нововъведенията му в последната версия. Едни от най-новите за нас неща, с които се сблъскахме, бяха процесите по автентикация и оторизация, моделирани с помощта на класа *JWT*. По отношение на някои елементи на езика намерихме сходство с други познати за нас технологии като *Java* и *C++*, които ни помогнаха да се ориентираме по-лесно в средата. В други случаи разчитахме на това сходство, а получавахме резултати, различни от очакваните.

Срещнахме нови неща по отношение на т.нар. *URL Rewriting* и как може да управляваме препращането към други ресурси с помощта на *.htaccess* файл.

За получаване на сполучливо уеб приложение нямаше как да не се наложи и интензивна работа с *JavaScript*. Научихме за възможностите, които езикът предлага за експорт към *xml* формат, както и затвърдихме работата по изпращане на заявки и комуникация с помощта на *бисквитки*.

По работата с *HTML* затвърдихме уменията си за вече познати елементи и се научихме да работим с елементи като *<dialog>*, които едва наскоро получиха масова поддръжка от всички популярни браузъри. Що се отнася до *CSS*, затвърдихме употребата на мощните инструменти за структуриране *CSS Flexbox* и *CSS Grid* и упражнихме използването на *CSS* променливи, с което вече имаме опит отпреди.

11. Използвани източници

[1] **Hollingworth**, Dave. *Create a PHP REST API* [online]. [<https://github.com/daveh/php-rest-api>]. [Публикуван: 2022-05-03]. [Последно посетен: 2023-06-23].

[2] **MoodleDocs**. *Moodle XML format* [online]. [https://docs.moodle.org/402/en/Moodle_XML_format]. [Публикуван: 2009-03-12]. [Последно редактиран: 2023-03-02]. [Последно посетен: 2023-06-23].

[3] **PHP** Documentation. *Function fgetcsv* [online]. [<https://www.php.net/manual/en/function.fgetcsv.php>]. [Последно посетен: 2023-06-23].

[4] **Web Dev Simplified**. *What is JWT and Why Should You Use JWT* [online]. [<https://www.youtube.com/watch?v=7Q17ubqLfAM>]. [Публикуван: 2019-07-27]. [Последно посетен: 2023-06-23].

Предал (подпис):

/62537, Стефан Велев, СИ, гр. П/

Предал (подпис):

/62547, Даниел Халачев, СИ, гр. П/

Приел (подпис):

/проф. д-р Милен Петров/