

**Факултет по изчислителна техника и автоматизация Катедра Компютърни системи и технологии**

**Курсов проект**

**Разработване на логическа уеб базирана игра**

Дипломант: **Петър Момчилов Тинчев**

Специалност: **Компютърни системи и технологии**

Факултетен номер: **21621396**

Адрес: [**http://lecode.byethost9.com/**](http://lecode.byethost9.com/)

Проверил:...............

/ас. Гинка Маринова/

**Съдържание**

**УВОД ................................................................................................................................... 3**

**ВЪВЕДЕНИЕ ...................................................................................................................... 5**

**2.1 АКТУАЛНОСТ НА ПРОБЛЕМА** .............................................................................................................................. 6

**2.2. ОБРАЗОВАТЕЛНИ ИГРИ** .................................................................................................................................... 7 **2.3 СЪЩЕСТВУВАЩИ РЕШЕНИЯ** ............................................................................................................................ 10

**2.4 ЕТАПИ ЗА РАЗРАБОТВАНЕ НА ЛОГИЧЕСКА ИГРА** .............................................................................................. 11 **2.5 НЕОБХОДИМИ ИНСТРУМЕНТИ ЗА РЕАЛИЗАЦИЯ** ................................................................................................ 11

*2.5.1.requestAnimationFrame ......................................................................................................................... 13* *2.5.2. Музика и звук ....................................................................................................................................... 13* *2.5.3. Офлайн игри и кеш ............................................................................................................................. 13*

*2.5.4.Индентифициране на играчите с BrowserID ................................................................................... 14* *2.5.5.Нива игрите ......................................................................................................................................... 14*

**ОПИСАНИЕ НА ИЗБРАНИТЕ СРЕДСТВА .................................................................... 15**

**3.1.ИЗПОЛЗВАНИ ТЕХНОЛОГИИ** ............................................................................................................................. 15

**3.2.HTML** ............................................................................................................................................................. 16

**3.3.CSS** ............................................................................................................................................................... 17 **3.4.JAVASCRIPT** ................................................................................................................................................... 18

**3.5.ANGULARJS** ................................................................................................................................................... 19

1. **6.D3 JS** ............................................................................................................................................................. 21 **3.7 PHP** ............................................................................................................................................................... 21 **3.8 MYSQL** ........................................................................................................................................................... 21

**ГЛАВА 4. ПРОЕКТИРАНЕ И РЕАЛИЗАЦИЯ НА ИГРАТА ........................................... 23**

1. **1.РАЗРАБОТВАНЕ НА WEB ИГРА** ......................................................................................................................... 24 4.2 ФУНКЦИЯ НА УЕБ БАЗИРАНАТА ИГРА “LECODE” .............................................................................................. 29 4.3 СЪХРАНЕНИЕ НА РЕЗУЛТАТА............................................................................................................................ 43
   1. АДМИНИСТРАТОРСКИ ПАНЕЛ ........................................................................................................................... 44
   2. КОЛЕЛОТО НА ЗНАНИЯТА ................................................................................................................................. 45 4.6 МАДЖОНГ КАРТИ ............................................................................................................................................. 48

**ГЛАВА 5. ТЕСТВАНЕ И НАСТРОЙКИ ........................................................................... 51**

1. 1 ТЕСТВАНЕ ...................................................................................................................................................... 52

**ГЛАВА 6 ЗАКЛЮЧЕНИЕ, АНАЛИЗИ, ВЪЗМОЖНОСТИ ЗА БЪДЕЩО РАЗВИТИЕ . 54**

1. 1ЗАКЛЮЧЕНИЕ .................................................................................................................................................. 54 6.2 ВЪЗМОЖНОСТИ ЗА БЪДЕЩО РАЗВИТИЕ ............................................................................................................ 55

**ГЛАВА 7 ИЗПОЛЗВАНА ЛИТЕРАТУРА ........................................................................ 56**

**ПРИЛОЖЕНИЕ ................................................................................................................. 58**

**Увод**

Компютърните игри са сред популярните забавления не само за младите, а и за хората в напреднала възраст. Историята на компютърните игри е част от историята на технологиите, изискването им е да работят с големи обеми от данни. Първоначално този тип игри са били разработвани и предназначени за военни и академични цели. С развитието на уеб технологиите жанра на браузър игрите набира голяма популярност. Статистиката показва, че в развитите страни 4/5 от населението до 35 години използват игри, а деца под 18 години всъщност почти всеки ден играят на компютърна игра. С тази популярност те могат да се използват при обучение, усвовяване и затвърдяване на знания в образованието. Полезните умения, които могат да бъдат да бъдат развити с помощта на компютърните игри са: изобретателност, вземане на решения, поставяне и преследване на цели, планиране и изпълнение на планове.

За първа компютърна игра се приема Spacewar!, разработена е през 1962 г. в Масачузетския технологичен институт. Spacewar! първоначално е стартирана на компютър с размери колкото един автомобил. Играта е реализирана с два космически кораба, летящи около една звезда, стрелящи се един друг. Целта е да се удари противника преди самоунищожение.

Първата комерсиална аркадна видео игра Pong е въведена 11 години след Spacewar!. Въпреки елементарната си концепция - бели правоъгълници на черен фон, тя получава световно признание.

Създаването на Space Invaders през 1977г. от Taito е класическа екшън игра.

Въведени са много елементи, които са стандарт във видео игрите.

Компютърните игри се сравняват като останалите видове игри - видео, конзолни и аркадни. Първите две категории обикновено се свързват с игри, които се пускат към телевизор, а аркадните са игри поставени на публични места.

Мултиплеърните игри придобиват популярност през 1993г., когато играта Doom се свързва на няколко компютъра.

Предимствата на компютъните игри пред останалите видове са в добрата графика, имат повече памет и скоростта на процесора е по-висока.. Мотивацията за реализиране на тази дипломна работа е насочения интерес, към създаване на игра използвайки нови технологии. Играта е създадена за студенти и фокусирана върху овладяване на учебното съдържание.

**Въведение**

Изследванията през годините в образованието сочат към непрекъснато намаляване на мотивацията и интереса към учебно-познавателната дейност.

Съвременната образователна практика е изправена пред предизвикателството на търсене на варианти за своето усъвършенстване, водещи до повишаване на мотивацията и интереса на ученици и студенти.

Обучението, базирано на игри може да се използва като форма за усвояване на нови знания и умения.

Целта на дипломната работа е разработване на логическа уеб базирана игра, насочена към потребители, които желаят да проверят своите знания в определена среда. За целта е избрана сферата на софтуерните технологии и по точно различните езици за програмиране. Студенти изучаващи компютърни науки имат възможност да затвърдят своите знания докато се забавляват. Играта има за цел да привлече вниманието им, чрез различния си интерфейс. и да затвърждават своите знания.

**2.1 Актуалност на проблема**

Компютърните игри заемат съществена част от свободното време на младите хора. Популярните жанрове на електронните игри са ролеви, стратегии в реално време, игри за стрелба и битка (екшън), приключенски, пъзел или шахмат. Обучението, базирано на компютърни игри понастоящем се прилага в различни професионални области: производството, енергетиката, компютърните технологии, висше образование, финанси, телекомуникациите, продажби, фармацевтика и услугите.

Всяка компютърна образователна игра се описва като се вземат под внимание нейните основни технологични характеристики:

* Цел
* Предназначение
* Развиваща функция
* Сценарии
* Нива на трудност.
* Отчитане на постигнати резултати.
* Положително стимулиране на играча

* Навигация
* Дизайн на интерфейса

На базата на посочените изисквания могат да се направят и класификации на игрите. Например по отношение на предназначението компютърните образователни игри могат да бъдат обучителни, свързани с определен учебен предмет или игри за свободното време с цел развлечение. Изключително важно за компютърната игра като средство за обучение е тя да притежава определена познавателна стойност, да се отличава с конкретни естетически качества, да бъде дружелюбна по отношение на играещия и да включва оригинална идея. От гледна точка като задание за разработване, компютърните игри трябва да позволяват надграждане на логическата структура.

**2.2. Образователни игри**

Разработката на уеб игри в днешно време е сравнително ново предизвикателство за програмистите. Въпреки това има и добри примери, от които могат да се вдъхновят. Безспорно едно от най-големите предимства на HTML5 игрите е, че те работят еднакво добре и на мобилни, и на десктоп устройства, използвайки един и същи код. Това означава, че ще се спести и време, и усилия, тъй като играта не трябва да бъде пренаписвана за различни операционни системи.

Една малка част от игрите са:

Планета 42 (фиг. 2.1) е уеб базарна образователна игра. Състои се от аркадни игри с елементи на знание. Интерактивни пъзели с научни данни, направени като обекти.

Предимства на играта са:

* безплатни тестове с картинки за оценка и самооценка.
* стартиране от мобилни утройства
* разработена е на два езика
* включване на картинки и анимации, което привлича вниманието
* представят знания, по забавен, интерактивен начин
* не изискват инсталация или регистрация

Недостатъкът на Планета 42 е ,че е разработена на Flash и за мобилните утройства е необходимо стартирането на .swf файлове.



**фиг.2. 1 Образователна игра Планета 42**

Bejeweled - Игрално студио PopCap създадена на HTML5 версията през 2011г. (фиг.2. 2). В играта е използван WebGL за да се осигури ускорение на графики, които попадат в canvas, ако WebGL не се поддържа. При играта целта е да се подредят три или повече еднакви фигурки по редове или колони. Bejeweled има добра графика, интересни аудио ефекти и много възможности за нива. Линк към играта: [www.popcap.com/sites/www.popcap.com/prettypkg/games/bejeweled2/flash/1033/bejeweled2.](http://www.popcap.com/sites/www.popcap.com/prettypkg/games/bejeweled2/flash/1033/bejeweled2.html)

[html](http://www.popcap.com/sites/www.popcap.com/prettypkg/games/bejeweled2/flash/1033/bejeweled2.html)



**фиг. 2.2 Играта Bejeweled**

Angry Birds - eдна от най-популярните игри в гейм пространството (фиг. 2. 3). Версия HTML5 е пусната през 2012 г., разработена с WebGL. Към днешна дата играта не е активна.



**фиг. 2. 3 Angry Birds -**

Runfield е създаден, за да покаже някои от възможностите на браузъра, като част от Firefox 4 през 2011 г., (фиг.2. 4). Той използва HTML5 canvas за графика и HTML5 аудио за звука. Недостатъкът на играта е липсата на някакво въведение за потребителя. Можете да играете тази игра, като посетите [https://developer.mozilla.org/en-](https://developer.mozilla.org/en-US/demos/detail/runfield)

[US/demos/detail/runfield](https://developer.mozilla.org/en-US/demos/detail/runfield) .



**фиг. 2. 4 Runfield**

**2.3 Съществуващи решения**

Обучението, базирано на игри позволява на учащите да предприемат задачи и опитни ситуации, които в друг случай биха били невъзможни или нежелателни като цена или време. Симулираната среда или реалистично пресъздаденият сюжет на ролева игра позволят на участниците да придобият опит в нещо, което е твърде скъпо, твърде рисковано и дори физически невъзможно за постигане в реалния свят. Симулациите и ролевите игри са в два жанра, ориентирани към забавление, но те са подходящи и за използване като инструменти и форми за обучение.

Развитието на информационните технологии спомага за изграждането на съвременни игри подпомагащи обучението. Логическите игри могат да са уеб базирани приложения в които се стартита игровото съдържание. При тях се използва архитектура от тип „клиент-сървър”, който става популярен през 80-те години на ХХ век. Клиентът и сървърът са компютърни програми като клиентът изпраща заявка към сървъра, която връща отговор на заявката. Играта е на сървъра, а потребителят се свързва с него чрез клиента или по-точно чрез браузъра от своя компютър. Тя е удобна за свързване на различни програми и потребители в една мрежа, но може да се реализира и на един компютър. Особеността на „клиент-сървър” технологията е, че в явна форма се допуска пространствена разпределеност между клиента и сървъра.

Предимства на архитектурата клиент-сървър са:

* даване на възможност на две или повече системи да си прехвърлят и споделят информация;
* позволява да се раздели потребителската информационна част от тази на преподавателя, като се въведат различни категории потребители за данните и тяхното генериране.
* позволява да се приложи специализация при създаването на ново учебно съдържание.

Недостатъци:

* сложна поддръжка на приложенията;
* необходимост от преинсталация на клиентските приложения при увеличаване на функционалността на приложението (мултимедия, динамика на изображенията).

**2.4 Етапи за разработване на логическа игра**

* анализиране на литературните източници и проучване на програмните системи и инструменти.
* определяне на целевата група, учебния материал.
* създаване на фонов сюжет, сценарий, проектиране на обекти, определяне на достъпа на играча.
* избор на технологични средства, програмни езици за разработване и реализиране на играта.
* създаване на работещ протип - обекти, анимации, звукови ефекти, съдържание, дизайн.
* тестване и дебъгване на завършения прототип отстраняване на грешки - логически, дизайнерски, технически.
* качване на играта в мрежата.
* тестване на реалния проект и отстраняване на грешки.

**2.5 Необходими инструменти за реализация**

Софтуерните технологии са едни от най-бързо развиващите се. За крайният потребител е създадена възможността да използва различни приложения улесняващи потребностите му. В Интернет пространството хората имат достъп до приложения предлагащи услуги, продукти, информация, обучение и забавление.

Към последната категория спадат уеб базираните игри. Те дават на потребителите бърз и лесен достъп, без инсталиране на допълнителни приложения, заемащи памет. Единствените необходими инструменти за стартирането им са браузър и Интернет. За да обхванат всички устройства, създателите на такъв тип игри трябва да оптимизират за различни браузъри и резолюции. При разработката им се използват понятия, като спрайтове (това са графичните елементи в играта), сцени или етапи, анимации, звук, 2D и 3D графики. С помощта на HTML5 canvas, SVG и WebGL се изобразяват 2D и 3D графики.

HTML5 canvas е JavaScript API, с възможност да се създават и редактират растерни графики в уеб средите. Силните страни са :

* интерактивност - може да отговоря на действията на потребителя.
* анимация - всеки един обект може да бъде анимиран.
* гъвкавост - разработчиците могат да създават всякакви елементи като линии, форми, текст, изображения и т.н., - с или без анимация, както и добавяне на видео и аудио.
* поддръжка за различните платформи -

HTML5 Canvas се поддържа от всички основни браузъри и от широка гама от устройства, включително компютри, таблети и смартфони.

Уеб стандарт - за разлика от Flash и Silverlight, canvas е отворена технология, която е част от HTML5, предлагаща безплатни и достъпни инструменти за разработване. Подходящ е за разработване на игри, реклама, декорации и обучения.

Координатна система в canvas - е 2D пространство, позициите се бележат с помощта на координатите X и Y. Оста X е по хоризонталата, оста Y по вертикалата, a центърът има позиция (0, 0). Този метод за позициониране на обекти, използван в математиката, е известен като Декартова координатна система.

Предимството на платното е, че може да произвежда точковидна пикселна графика, а недостатъците са, че производителността му е различна при различните платформи и функционалността на анимацията не е вградена. От друга страна SVG (Scalable Vector Graphics — мащабируема векторна графика) е друго решение за 2D графиките. Той може да съдържа векторна графика, растерна графика, текст, цвят, градиент и др.. Предимствата на SVG са, че имат възможност за вградена анимация и самите графики могат да се мащабират лесно за различни устройства и резолюции.

Негативният аспект е, че не е толкова популярен, колкото HTML canvas.

WebGL или Web Graphics Library e JavaScript API за интерактивана 3D графика. Базиран е на OpenGL ES 2.0 и осигурява цялата функционалност, необходима за производство на някои зрелищни ефекти. Предимството на WebGL е, че дава възможност за сложни визуализации, а негативите са, че е сложен за изучаване и все още не се поддържа от Internet Explorer (IE).

### *2.5.1.requestAnimationFrame*

Повечето анимации в web приложенията са създадени за да се променя движението на екрана. Често се използват JavaScript функции като setTimeout() или setInterval(), които водят до намаляване на производителността на приложението. За да се разреши този проблем се въвежда методът requestAnimationFrame. Целта му е да дава контрол върху анимацията, като увеличава производителността и предотвратява евентулани проблеми. Предимството на requestAnimationFrame е, че при неактивен таб на браузъра анимацията спира да се изпълнява в сравнение с setTimeout() и setInterval(), при които независимо дали браузъра е минимизиран, или таба е неактивен анимацията е активна.

#### 2.5.2. Музика и звук

Добавянето на звук е един от основните компоненти в игрите. До скоро прилагането на аудио е било възможно чрез Flash, но се поява елемент HTML5 <audio>. Той се използва за единични звукови файлове, като фонова музика в рамките на играта. Аудио елемента не е предназначен за звукови ефекти, които често са в бързо темпо и трябва да се изпълняват едновременно повече от един звук. Решението за това е въвеждането на Web Audio API за Chrome и Audio Data API за Mozilla. Чрез тях могат да се създават звуци от JavaScript, да се редактира аудио и да се извличат данни в реално време.

Звуковите ефекти в web средата търпят развитие.

#### 2.5.3. Офлайн игри и кеш

Интернет игрите са интересни за потребителите, но какво ще стане ако пожелаят да играят офлайн, или Интернет връзката прекъсне по-средата на гейм сесията. Сегашните уеб игри спират веднага след като спре Интернета, не се изпращат заявки към сървъра и не се запазват данните на играча.

За да се оптимизира този процес се добавя кеш, който запазва конкретните активи на потребителя. Друг вариант е локално съхранение на данните на играча и периодично синхронизиране със сървъра.

Използват се Local Storage и IndexedDB за съхраннение на много мегабайта данни. Също така може да се добави функционалност, която да предупреждава играча за неговото online състояние. Чрез свойството navigator.onLine което засича статуса, може да се задейства автоматично поведение на играта.

### *2.5.4.Индентифициране на играчите с BrowserID*

iOS предлага услуги като OpenFeint и Game Center на Apple, които идентифицират играчите. За Mozilla това става с помощта на BrowserID. Тази услуга предлага един единствен вход за уеб приложенията, свързани чрез е-мейл адрес (с възможност за добавяне на повече от една електронна поща). Идеята е, че винаги се помни имейл адреса на потребителя, вместо измислено потребителско име или URL.

Опорните точки на BrowserID са:

* лесен за употреба
* сигурност
* изпълнение крос-браузър
* Децентрализирано валидиране

### *2.5.5.Нива игрите*

Разглеждаме различните видове игри и как те са изградени. Основно са два вида, когато става въпрос за браузърни игри:

* основаващи се на определени нива, тип скочи и бягай, Angry Birds.
* автоматично генериране на нива или създадени от потребителя като част от играта.

Причината да се манипулират по този начин игрите е да стават популярни, интересни за играчите, за да се връщат отново и прекарат колкото се може по-дълго време в играта. Колкото повече предизвикателства има една игра толкова повече хора я споделят и използват.

**Описание на избраните средства**

**3.1.Използвани технологии**

През последните години се наблюдава мощно развитие на уеб технологиите, включително развитието на двигатели като WebKit, Gecko, JavaScript двигатели V8, SpiderMonkey, както и появата на новия HTML5 API и CSS3. Тези стандартни технологии са на път да заменят RIA (Rich Internet Application) плъгините като Flash и Silverlight. При тези технологии, се търси лесен начин за интегриране на приложенията за всеки магазин (Android Market, Apple Store, Chrome Web Store). Разработват се уеб приложения, които са оптимизирани за различните платформи. От друга страна браузърите също имат своите ъпдейти, с които трябва да се съобразим. Към днешна дата са известни много библиотеки, рамки, плъгини и инструменти, създадени за да улеснят работата на програмистите.

Проследявам развитието на познатите езици за създаване на уеб приложения.

# 3.2.HTML

HyperText Markup Language е основният маркиращ език за описание и дизайн на уеб страници. HTML5 е новата версия, която включва много на брой функции, които правят уеб технологиите отворен стандарт с безкрайни възможности, съчетаващи HTML, CSS и JavaScript. В тази версия са включени тагове като video, audio, canvas, section, nav и др., както и интеграцията на SVG съдържание, офлайн съхранение на база от данни (офлайн уеб приложения), влачене, cross-document съобщения, управление на историята в браузъра, microdata, геолокация и др. Някои елементи от HTML 4.01 са премахнати като font и center.

HTML5 е проектиран така, че старите браузъри да могат да игнорират новите му конструкции. За разлика от HTML 4.01, новата HTML5 спецификация задава ясни правила за обработка и анализ, с цел да осигури един и същи резултат от различните по съвместимост браузъри в случай на неправилен синтаксис. Въпреки, че HTML5 въвежда съвместимост с документите с неправилен синтаксис (т. нар. "tag soup" документи), тези документи се определят като неотговарящи на HTML5 стандарта.

# 3.3.CSS

Cascading Style Sheets език за стилизация на HTML таговете. Чрез него освен задаване на цвят, размер, позиция, видимост, и прозрачност, могат да се анимират елементи. Тази опция идва с най-новия стандарт CSS3, включващ селектори, текстови ефекти, 2D и 3D трансформации, боксови модели, грид и др. Както е показано от примера по-долу, анимация може да се направи чрез @keyframes (ключова дума). Задава се име на анимацията “animationFrames” и във фигуралните скоби се описва в проценти стъпките на изпълнение от начално състояние до краен резултат. Анимацията може да се разбие на малки интервали, като се добавят повече breakpoints (0%, 25%, 50% и т.н.). Други настройки, които могат да се добавят са: време за изпълнение, време за изчакване, колко пъти да се повтори, стилове които да се анимират - цвят, позиция, размер, видимост, завъртане, разпъване, увеличаване и др.

.element-animation{ animation: animationFrames linear 4s; animation-iteration-count: 1;

}

@keyframes animationFrames{

0% { transform: translate(0px,0px) rotate(0deg) ; }

100% {transform: translate(200px,0px) rotate(180deg) ; }

}

За да се изпълни анимация чрез @keyframes на всички браузъри е необходимо използването на префикс.

* *“-webkit” - Safari, Chrome*
* *“-moz” - Firefox*
* *“-ms” - Internet Explorer*
* *“-о” - Opera*

Голяма част от стиловете в CSS3 използват префикс.

CSS Transitions - друг начин за лесно добавяне на анимации, който дава възможност да се променят CSS стойностите плавно през определен период от време. За да се използват Transitions първо трябва да се определят състоянията и действията.

Свойствата които протежава са:

* transition-property: описват се свойствата, които ще се анимират, по подразбиране е all;
* transition-duration: задава се време на изпълнение на прехода;
* transition-delay: задава се време преди да започне прехода;
* transition-timing-function: задава се скоростта, с която да се изпълни прехода; CSS Transitions могат да се използват при клик, ховър, скрол, мърдане на мишката и други събития създадни от потребителя.

CSS Transforms - позволява елементи да бъдат завъртани, мащабирани, и изкривени в 2D пространство. Има и CSS 3D трансформиране, но все още не е приложим навсякъде. CSS 2D Transforms имат две свойства:

* transform-origin - помага за откриване на центъра на трансформация;
* transform - позволява да се прилагат трансформации на избрани елементи. Четири са значимите трансформиращи функции: translate(x, y), scale(x) or scale(x, y), rotate(angle), and skew(angle);

CSS3 въвежда няколко нови единици, установяващи съотношенията между височина и ширина. Това касае пряко размера на даденото изображение, визуализирано от различните устройства. Новите единици са vw (широчина на прозореца), vh (височина на прозореца), Vmin ( минимална височина и ширина на прозореца) и Vmax ( максимална височина и ширина на прозореца). С течение на времето, ролята им е важна в респонсив дизайна. Той от своя страна се стреми да не използва фиксирани единици, като пиксел и инч, тъй като всяко устройство е с различни параметри. Това налага адаптиране на сайта, към изискванията на всеки вид устройство.

Използването на CSS стила flex помага за лесното изграждането на грид в една страница. Той определя дължината на елемента, в сравнение с останалите елементи в същия контейнер. Има опцията за създаване на редов или колонов грид. Поддържа се от всички браузъри, като за някои от тях са необходими допъллнителни настройки.

# 3.4.JavaScript

Поддържа обектно-ориентиран и функционален стил на програмиране, позволява създаването на динамични уеб страници и приложения.

Езикът се зарежда, интерпретира и изпълнява от уеб браузъра, който му осигурява достъп до обектния модел на браузъра. Основни задачи, които изпълнява са: зареждане на данни с AJAX, използване на камера и микрофон, създаване на 3D графики, разпознаване на операционни системи и браузъри, ефекти с изображения и HTML елементи: скриване/показване, пренареждане, влачене, слайд шоу и др. При използването на JavaScript или на някои от неговите бибилиотеки, трябва да се внимава при извършване на математически изчисления, защото е слабо типизиран език и възникват грешки.

Разнообразието от библиотеки се увеличава в уеб общността. Някои от известните и влиятелни рамки са: AngularJS, React, Backbone и Ember. Всяка една от тях помага на програмистите да създават приложения богати на функционалност и производителност.

# 3.5.AngularJS

JavaScript framework създаден от Google, работи на принципа Model View Controller, различава се с това, че шаблона и данните се доставят на браузъра, за да бъдат сглобени в него, ролята на сървъра е само да служи като статичен ресурс. Библиотеката първо прочита HTML страницата, в която има вложени допълнителни специални тагове и атрибути. Тези атрибути се интерпретират от AngularJS като директиви сочещи дали да бъдат визуализрани входящи или изходящи части на страницата от модела представен от страндартни JavaScript променливи. Стойностите на тези JavaScript променливи могат да бъдат зададени в кода или извлечени от статични или динамични JSON ресурси.

Основната концепция, на която е изграден AngularJS е model-view-controller (MVC) архитектурният модел, чиято идея е при разработването на големи приложения информацията да се раздели на логически дялове. MVC моделът разделя уеб приложението на три отделни части.

Модел (Model) – основно включва информация относно данните в приложението, която обикновено се взема от сървъра (например данните на един студент в университет: име, презиме, фамилия, ЕГН, факултетен номер и т.н.)

Изглед (View) – представлява информацията, която се визуализира на екрана. (В примера това може да бъде името, фамилията, факултетният номер, а останалите данни да остават скрити за потребителя).

Контролер (Controller) (също известен като viewmodel или presenter) представлява софтуерният код, който осъществява връзката между данните, които се пазят на сървъра (Модел) и това кои от тях да се визуализират на екрана (Изглед).

Основните понятия в AngularJS са:

* Scope – контекстът, където моделът се съхранява, така че контролерите, директивите и изразите да могат да го достъпват;
* Directives – разширяват HTML с допълнителни потребителски атрибути и елементи
* Model – данните, които потребителят вижда в Изгледа, и с които обменя информация
* Service – бизнес логика, която се използва многократно (независимо от Изгледа)
* Module – контейнер за различните части на приложението, включва контролери, сървиси, филтри, директиви, които конфигурират Injector.

В приложенията написани с Angular, визуализацията е DOM дървото, контролерите са JavaScript класове, а данните на модела се съхраняват в свойства на обектите.

Архитектура:

* Two-way binding - всички промени в изгледа (view) са веднага отразени в модела (model), както и данните от модела (model) се отразяват веднага на изгледа (view).
* Dirty checking - проверя автоматично дали има нещо променено в DOM дървото;
* Dependency injection - зависимо инжектиране, с който се проверява как компоненти се сдобият с техните зависимости.

Другата сила на AngularJS е маршрутизирането между страниците без презареждане на потребителската сесия, наречено SPA приложение (Single-page application). Последните две години SPA приложенията навлязаха с пълна сила в уеб технологиите. Те се отличават със способността си да преначертаят всяка част на потребителския интерфейс, без да изисква от сървъра двупосочно извличане на HTML.

# 3.6.D3 JS

JavaScript библиотека за визуализиране на данни чрез използване на уеб стандартите SVG, Canvas и HTML. Съчетава мощни визуализация и техники за взаимодействие с данни. Тя не е монолитна рамка, имаща за цел да осигури всички възможни функция, вместо това решава същината на проблема чрез ефективно манипулиране на документа, основаващ се на данни. D3 е изключително бърза, поддържаща големи масиви от данни, динамични Hover ефекти и анимации.

Чрез нея лесно могат да се създадат и визуализират векторни изображения на различните видове диаграми, графи и др. Версия 4.0 е колекция от модули, направени да работят заедно.

## 3.7 PHP

Hypertext Preprocessor скриптов език с отворен код за създаване на сървърни приложения и динамично уеб съдържание. Синтаксиса му е базиран на C и Perl. Богат е на математически функции, работа с низове и регулярни изрази, изпращане на форми, работа с бази данни, работа със сокети, управление на сесии и работа с бисквитки. Това, което отличава PHP от клиентския JavaScript, е че кодът се изпълнява на сървъра, генерира HTML и изпраща към клиента. Клиентът получава резултата от изпълнението на този скрипт, без да знае какъв код е генерирал този резултат. Към днешна дата версията e 7.0.5. Фокусът ѝ е в премахване на (deprecated) изоставени функционалности от предишни версии, подобряване на консистенцията на самия език и подобряване на производителността му.

## 3.8 MySql

Многопотребителска SQL (Structured Query Language) система за управление на бази данни. Поддържа интерфейси за програмиране под множество езици - C/C++, Eiffel, Java, Perl, Python, но често се използва в комбинация с езика PHP. Данните, съхранявани в MySQL могат да бъдат както обикновени текстови списъци, така и сложна структурирана информация. Управлението на системата най-често се осъществява чрез инструмента phpmyadmin. Също така има и приложения с които могат да се управляват данните локално.

В уеб игрите MySql може да съхранява информация на всеки един регистриран потребител, данни за различните нива, динамично сменящи текстове и др.

Основни причини, поради който избрах MySQL са:

**Скалируемост и гъвкавост** – MySQL предоставя възможност за дълбоко вграждане в приложенията с отпечатък само 1MB за използваните масивни хранилища от данни. Подържа се от всички платформи Linux, Unix и Windows. Понеже е с отворен код може да се персонализира според изискванията на клиента. Висока производителност – MySQL е подходяща както високоскоростни транзакционни приложения, така и за сайтове с голям обем от данни и заявки над милиард на ден.

**Висока надеждност** – за да гарантира високата надеждност, MySQL предлага различни инструменти за подсигуряване като: касетъчни сървъри, високоскоростни репликации и др.

**Поддръжка на транзакции** – MySQL предлага една от най-мощните транзакционни системи на бази данни на пазара. Характеристиките включват неограничено ниво на заключване на редовете ACID транзакции (atomic, consistent, isolated, durable).

**Атомарност** – изпълняват се или всички инструкции в дадената транзакция или не си изпълнява нито една.

**Консистентност/съгласуваност –**  Информацията в базата данни оставалогически коректна постоянно. Тя преминава от едно консистентно състояние в друго консистентно състояние.

**Изолираност** – транзакциите са изолирани една от друга. Ако две транзакции се опитват да правят промени по една и съща таблица, то първо ще мине едната, после другата. **Издръжливост** **–** ако една транзакция е потвърдена (committed), то тя не може да бъде загубена.

**Силна защита на данните** – MySQL предлага мощни механизми за осигуряване на достъп само на оторизирани потребители. Осигурени са също SSH и SSL поддръжка за осигуряване на безопасни и сигурни връзки. Съществуват редица методи за ограничаване на данните за потребителите, за да виждат само необходимата информация. Също така поддържа пълно архивиране и възстановяване на данните по лесен начин.

Лесно управление и поддръжка 24x7.

**Проектиране и реализация на играта**

**4.1.Разработване на web игра**

Изработването на софтуер изисква умения, креативност, опит, работа в екип и желание за създаването добър продукт. Всеки разрабочик на подобни приложения обръща внимание както на функционалността, така и на UE & GUI. Първото нещо което вижда потребителя когато използва подобен софтуер е интерфейса. Той трябва да бъде приятен и удобен.

Подборът на правилен жанр за игра е един от важните моменти при създаването ѝ. Игрите от всеки тип жанр могат да се използват за обучение или тренировка на уменията. За различните цели на обучението и възраст на целевата група от обучавани обаче са подходящи различни типове игри. Накратко популярните от тях са:

Пъзел-игри: В тази група се включват пъзели с разместване на движещи се части; търсене на разлики между две почти еднакви картини; маджонг – търсене на двойки еднакви плочки от множество, разположено в определена конфигурация; игри за запаметяване на местоположението на обекти; търсене на предмети в някаква картина и други. Подходящи са за по-млади играчи и могат да се използват за почти всяка учебна област.

Ролеви игри: играчът играе роля в някакъв сценарий или ситуация. При тях може да има други играчи или да се играе самостоятелно. Когато се играе от повече потребители, те могат да бъдат съюзници или съперници помежду си или да бъдат разделени на отбори. Подходящи са за професионално обучение и могат да се изпълнят под формата на симулатори.

Игрите, базирани на ролевия жанр заемат място в реалистичен свят в специфично време (в средновековието, в настоящето или в бъдещето) или в измислен свят, който е или не е близък до реалността. Играчите са представени от и контролират аватар (найчесто в човешка форма) и решават множество различни задачи в продължение на играта. Това налага играчът да намира определени обекти, от които се нуждае, за да реши даден проблем и/или се изисква правилен отговор от няколко дадени отговори на конкретен въпрос. Задачите са свързани с играчите чрез неигровите персонажи (NPCs – Non-player Characters), които не могат да бъдат контролирани от играча като той взаимодейства с тях чрез диалози.

Приключенски игри: Играчът се движи в поле, като има някаква цел. За да достигне до нея, той трябва да изпълни множество от по-малки задачи.

Тези задачи могат да включват елементи от останалите жанрове игри (разговори, битка, решаване на логическа задача). Подходящи са за всяка възраст и могат да бъдат използвани при разнообразни учебни области. Причината за тази универсалност е, че при този тип игри могат да се проектират и влагат много различни ситуации и игрови обекти.

Рали: Потребителят шофира автомобил (в различните вариации може да бъде заменен с мотоциклет, велосипед, тежкотоварен камион или друго превозно средство). Целта може да е да финишира първи или да премине успешно през дадено трасе. При реализацията като симулатор този тип игри могат да се използват за тренировка при шофьорски курсове, както и при обучение или изпит върху познаване на правилата за движение.

Стратегии: Развиване на обект (ферма, град, царство, собствен бизнес). Може да се взаимодейства с други играчи. Според типа на играта потребителите може да са съюзници или врагове. Подобно на приключенските игри, стратегиите са подходящи за широка възрастова група и за много различни области.

Логически: Създаване на път (разчистване на път, строене чрез поставяне на обекти по определен начин, свързване на 2 точки, постигане на цел чрез взаимодействие с различни обекти или отговаряне на въпрос). Тези игри са подходящи са за всички възрасти и могат да тренират и възпитават качества като логическо мислене и наблюдателност.

Игри с карти: стандартни игри или пасианси. Този тип игри не са особено подходящи за обучение на подрастващи. Въпреки това биха могли да намерят приложение при обучение на деца с проблеми, свързани с възприемане на символи.

Хазартни игри: рулетка, слотове, ротативки, покер, зарове. Поради твърде силното влияние на случайността, тези игри не са подходящи за учебни цели. Елементи от тях, обаче, могат да се включат в други игри с цел вкарване на елемент на случайност в тях.

Образователни игри: Тези игри имат за цел предаването на специфични знания или умения на обучаемите.

Проектирането на компютърна игра е сложен процес, тъй като включва проектирането на цялото преживяване на играча като интерактивност, сюжет, аудио и визуални ефекти.

Включва също много повече компоненти, които са съчетание на функционалност и визия, за реализирането на които се изискват и артистични нагласи и умения. Разработването им е интензивен творчески процес през цялото времетраене на проектирането. То изисква интегриране на усилията и сътрудничество между различни професионални екипи в сравнение с тези за други видове софтуерни системи. Създаването на игри има мултидисциплинарна природа, защото в допълнение към софтуерното развитие включва области като игрово проектиране, проектиране на нива и сюжет, графичен дизайн и изпълнение, звуково инженерство, комерсиализация. Проектирането като концепция може да бъде дефинирано като план или модел на това, което трябва да бъде направено. Разработването на обикновено изисква екип от разработчици, който може да бъде многоброен. Един такъв екип събира на едно място проектанти (дизайнери), различни програмисти, специалисти в областите на графиката, изкуствения интелект, осигуряване на качеството, режисьори, хора на изкуството и музикални техници. Организират се срещи в които се планират и обсъждат различни задачи, проблеми които могат да възникнат, разбиване на проекта на отделни задачи (спринтове), задване на време изчислено на база брой хора, часове и работа.

През последните години гейм индустрията е източник на най-добрите софтуерни технологии на пазара. Генераторите на 3D сцени и гейм машини превъзхождат по представяне и параметри програмите за изпълнение на приложения, създадени в сериозната индустрия. Фирмите които разработват такъв софтуер са приели принципите на интерфейса човек-машина и ефективен графичен интерфейс. Игрите са отразили основните особености на човешкото поведение, а удобният интерфейс създава възможност на хората да използват приложенията на потребителското устройство.

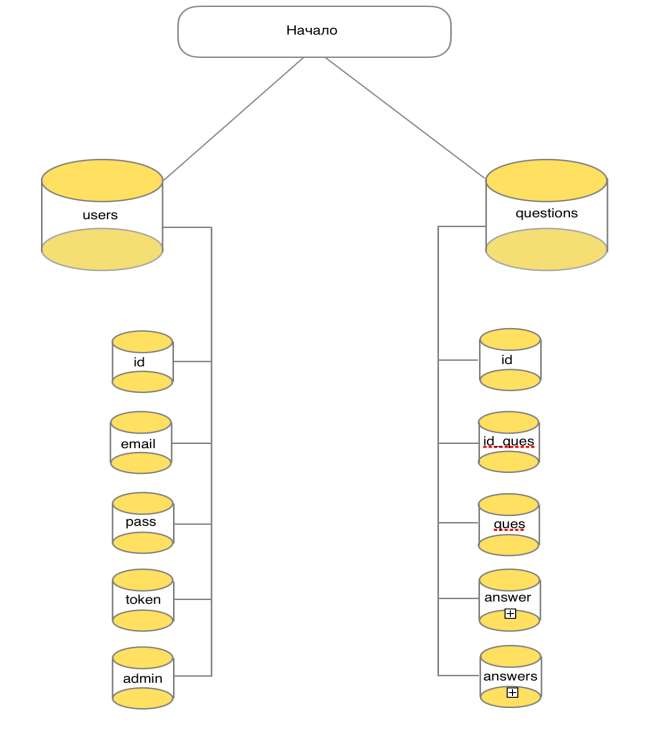
Комбинацията на мотивацияс учебна методология, прави обучението бързо и ефективно.

Изключително важно за компютърната игра като средство за обучение е тя да отговаря на следните условия:

* да притежава определена познавателна стойност;
* да се отличава с конкретни естетически качества;
* да бъде дружелюбна по отношение на играещия;  и по възможност да включва оригинална идея.

Управление на системата.

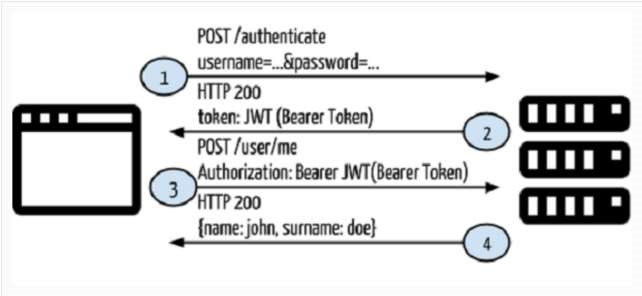
Блокова схема на разработената “LECODE” игра.



**фиг. 4.1 Блокова схема на системата**

Първата колона включва информация за потребителите като идентификационен номер, имейл, криптирана парола, жетон(token) и колона администратор, в която се записва дали потребителя има администраторски права. При регистрация се добавя жетон, чрез който може да се влезе в системата. Ролята на жетона е да провери потребителя дали съществува в базата данни. Потребители без регистрация не могат да се логнат в играта. Идентификацията е важна част на всяко уеб приложение.

Token-базирано идентифициране (фиг. 4. 2.) :



**фиг. 4. 2 Token-базирано идентифициране**

При този тип идентифициране не се използват сесии и бисквитки. Жетона идентифицира потребителя с всяка заявка към сървъра. Потребителят предоставя име и парола във формата, след като заявката е направена се валидират данните, ако те са коректни се създава жетон използван за информация. Тогава се връща тази информация като отговор в заглавната част, която има възможност да се запази като символ на браузъра и да се използва за локално.

Втората колона съдържа информация за въпросите като идентификационен

номер на самата таблица, уникален номер на въпроса, самия въпрос, номер на правилния отговор на въпроса и различните типове отговори.

Последната колона с видовете отговори, всеки един от тях съдържа в себе си идентификационен номер, ключ и стойност. Чрез предаване на идентификационен номер се осъществява задаването на правилният отговор в базата.

Базата от данни е създадена чрез системата phpMyAdmin. Не е необходимо да се използва специализиран софтуер при работа с даните, а за администраторски панел се ползва същата система с функционалностите, които позволява системата.

phpMyAdmin е уеб базиран инструмент за администрация на MySQL бази данни. С негова помощ могат да се извършват различни действия като създаване и изтриване на база данни, променяне на таблици, добавяне, изтриване и редактиране на полета, изпълнение на SQL код и управление на ключове. Той поддържа широк спектър от операции на MySQL и MariaDB. Характеристики на phpMyAdmin:

* Интуитивен интерфейс
* Импортиране на данни от CSV и SQL
* Експорт на данни към различни формати: CSV, SQL, XML, PDF, ISO / IEC 2630, OpenDocument Текст и електронна таблица, Word и други
* Администриране на множество сървъри
* Създаване на графики на оформлението си база данни в различни формати
* Създаване на сложни заявки, използващи Query-по-пример (QBE)
* Трансформиране на съхраняваните данни във всеки формат с помощта на набор от предварително дефинирани функции, като показване на BLOB-данни като изображение или линк.

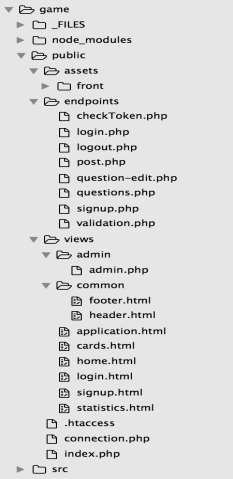
**4.2 Функция на уеб базираната игра “LECODE”**

Разработената игра е за проверка на знанията в областта на софтуерните технологии. Името на играта е съчетане от две думи “Learn” и “Code”. Съдържанието на играта се състои от две нива.

В първото ниво играчът отговаря на въпроси свързани с част от различните

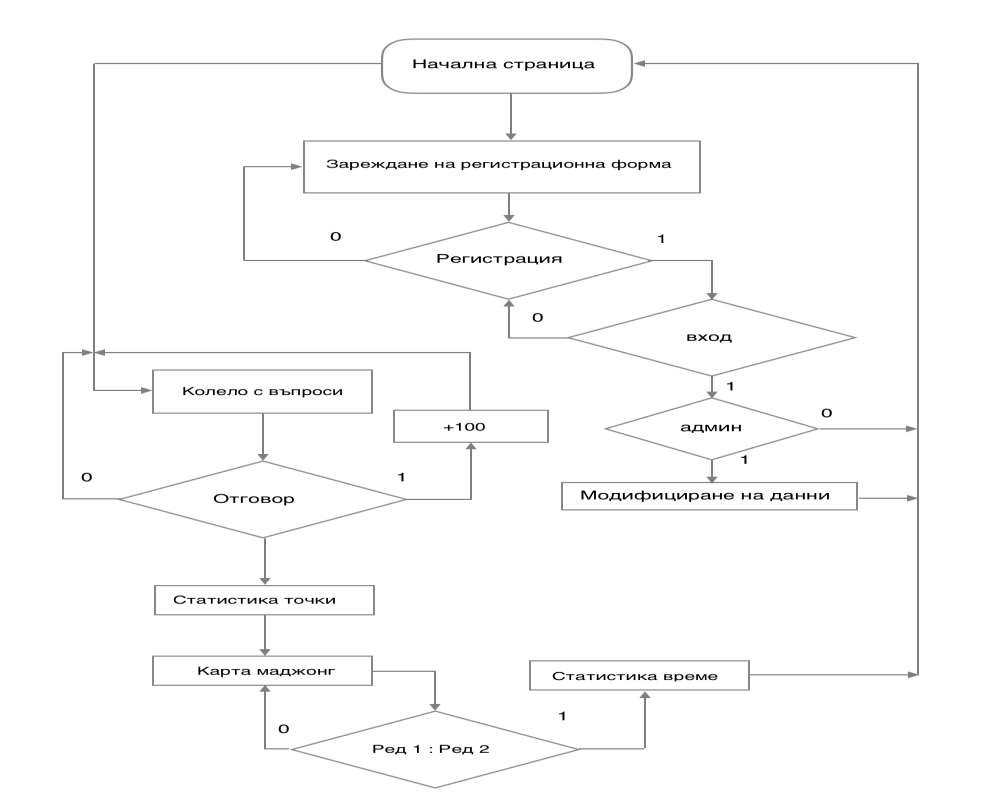
езици за програмиране. Визуализира се на екрана въпрос с три опции за отговор като само един от тях е верен. Играчът няма да бъде допуснат до следващ въпрос ако не отговори на вече активирания. Всеки верен отговор носи по 100 точки.

Ако играча не отговори правилно, точките не се променят. По този начин им се дава стимул. Второто ниво от играта е за време. То се активира само ако играчите са преминали първо ниво. На екрана се зареждат два реда с карти маджонг. Целта е за кратко време играчите да намерят двойките карти. Единия ред съдържа наименованието на език за програмиране, а вторият ред съдържа извадка от код на съответния език. Играта е реализирана с администаторски панел, чрез който може да въвеждат и управляват данните. Панелът има опции за добавяне, редактиране и изтриване на въпросите от първо ниво. На фиг. 4.3. е показана структурата на файловете:



фиг. 4. 3 Структурата на файловете

На фиг. 4.4 е представено функционалното описание на играта.



фиг. 4. 4 Функционалност на играта

В разработената игра “LECODE” базата данни съдържа две таблици (таб. 4.1. и таб.

4.2).

Таблица съхраняваща регистрираните потребители (таб 4.1).

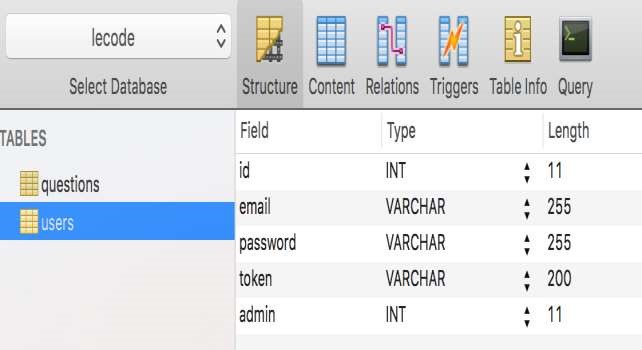


таблица. 4. 1

Таблица съхраняваща въпросите и техните отговори (таб 4.2).

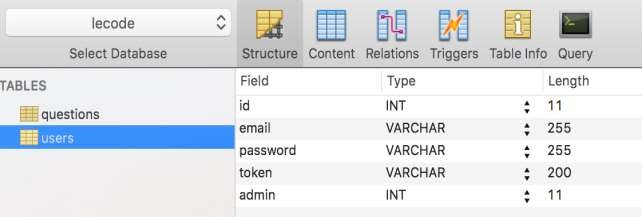
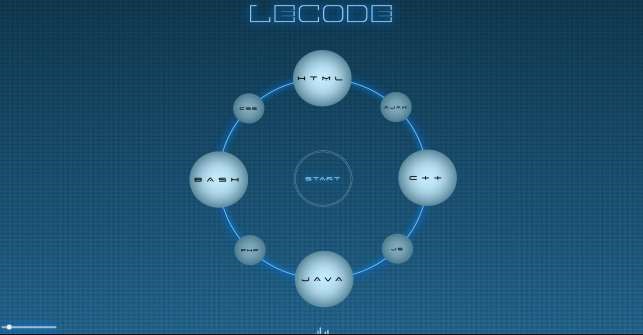


таблица. 4. 2

В началната страница на играта са визуализирани осем различни категории от езиците за програмиране. Те са абсолютно позиционирани елементи движещи се по посока на часовниковата стрелка.(фиг. 4.5). Като настройка играча има възможността да регулира силата на звука, който е като фон на играта.



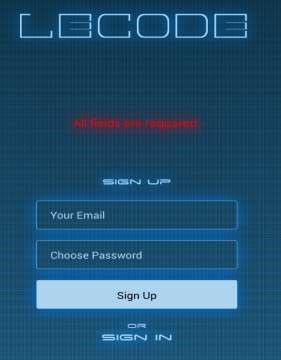
фиг. 4. 5

След натискане на бутона старт се зарежда страницата с формитите за регистрация и логин. (фиг. 4.6 и фиг. 4.7). Потребителите имат възможността да се регистрират чрез имейл и парола или чрез потребителско име и паррола.



фиг. 4. 6 фиг. 4. 7

Полетата във формите са задължителни и имат съответната валидация дали потребителя съществува в базата данни. (фиг. 4. 8 и фиг. 4.9). Ако някое от полетата в логин формата не са въведени правилно, ще се покаже съответното съобщение за грешка.



фиг. 4. 8 фиг. 4. 9

Запазването на новите потребители и тяхното съхранение е реализирано с PHP и по-точно PDO. Разширението на PHP Data Objects определя лек, последователен интерфейс за достъп до данните. PDO може да използва Prepared Statements, това означава, че първо подготвя заявката като създава подобие на шаблон и след това я попълва и изпълнява, като по този начин изгражда една много силна защита от вида SQL Injection атаки. При използването му няма нищо различно от обикновения синтаксис. За осъществяване на връзка с базата данни се създава клас connection. Задават се необходимите данни за връзката като хост, име на БД, порт, потребителко име и парола.

*$db = new PDO(“mysql:host=localhost;*

*dbname=dataase\_name; port=2222”,*

*"username", “pass"*

*“);*

Идентифицирането на потребителите е разработено чрез AngularJS Authentication

Service. Той съдържа методи за идентифициране, определяне на идентификационните данни и изчистването им от HTTP "Authorization" headers използващи AngularJS $http service за ефективно логване и разлогване. В този случай методът "service" раздава жетони на упълномощените потребители.

При post заявката се зарежда файла checkToken.php, който добавя token в базата.

*app.service('AuthenticationService', ['$http', '$state', function($http, $state){ var self = this; self.checkToken = function(token) { var data = {token: token};*

*$http.post('endpoints/checkToken.php', data).success(function(response){ if (response == 'unauthorized')*

*$state.go('login');*

*else*

*return response;*

*}).error(function(error){*

*$state.go('login');*

*});*

*}*

*}]);*

Друг интерсен метод използван чрез Angular е маршрутизирането на страниците без робновяване на браузъра. Config метода към модела арр съдържа функция с два параметъра $stateProvider и $urlRouterProvider. $urlRouterProvider съхранява в себе си главния url на проекта “domain.com”, като в края се задава “/”. Ролята на $stateProvider е след “/”. Той управлява различните състоянията (state) в адрес бара.

Във всеки state може да се зададе custom url, controller - активен за текущата страница, templateUrl - адреса на странцата която да зареди, animation - създаване на клас за анимация при смяна на страниците. Чрез добавянето на контролер в state може да се ограничи достъпа на на потребителите към някой страници. В примера е използван MianController, който следи дали потребителя е логнат. При идентификация MianController дава достъп до останалите страници, в противен случай запазва текущото си състояние.

Работи по следният начин: при състояние login, взема файл с url views/login.htm, зарежда контролер LoginController, анимира страницата с клас animate-page и слага към

“domain.com” -“/login”.

*app.config(function($stateProvider, $urlRouterProvider){*

*$urlRouterProvider.otherwise(‘/');*

*$stateProvider*

*.state('login', {*

*url:'/login',*

*controller: 'LoginController', templateUrl: 'views/login.html', animation: 'animate-page'*

*})*

*.state('application', {*

*url:'/арр',*

*controller: 'MainController', templateUrl: 'views/application.html', animation: 'animate-page'*

*})*

*………..*

*.state('index', {*

*url:'/',*

*templateUrl: 'views/home.html', animation: 'animate-page'*

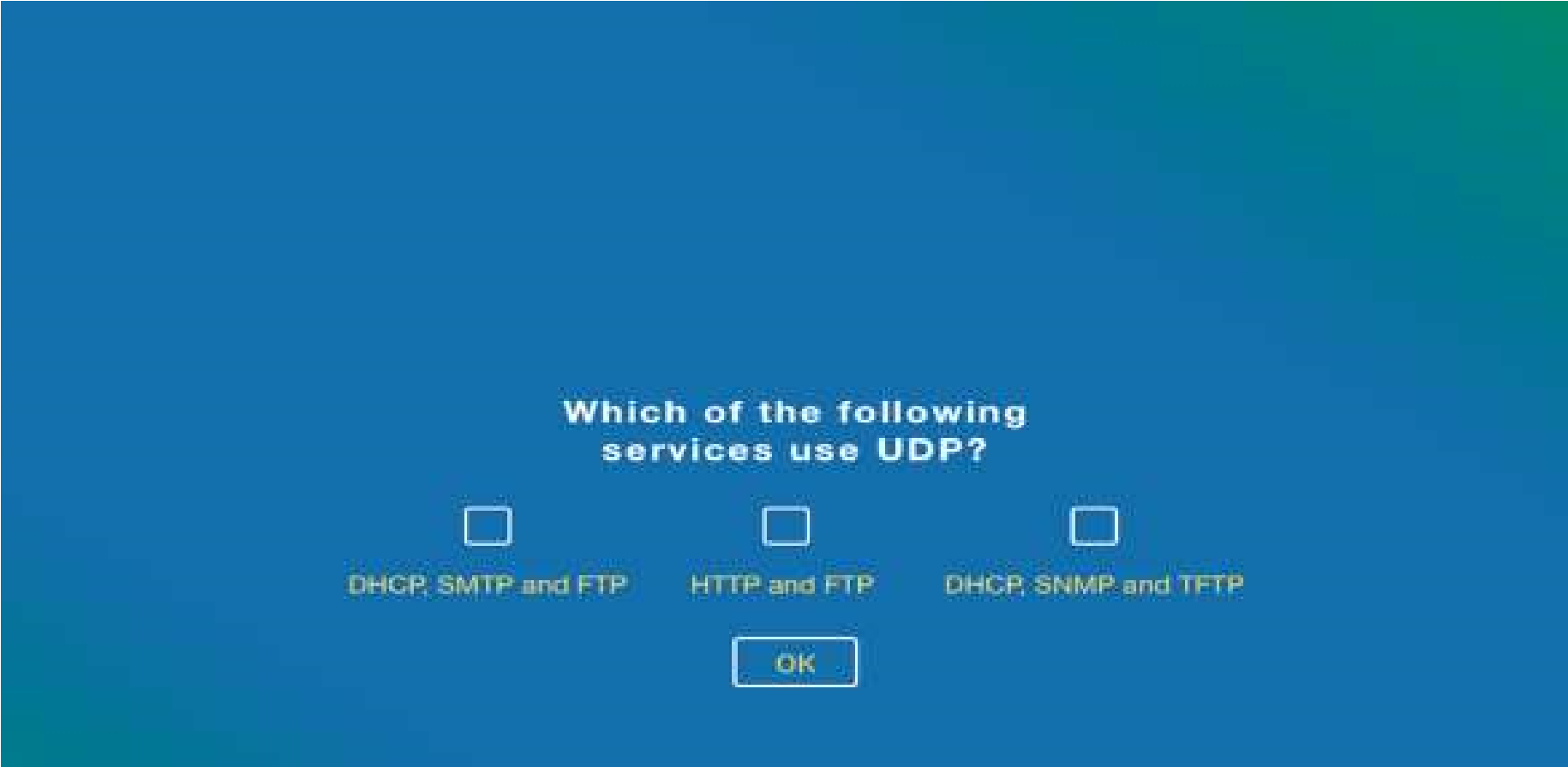
*})*

*});*

Когато state е активиран, шаблона автоматично се вмъква в ui-view, което е главното състояние намиращо се в DOM дървото. Атрибута ui-view се поставя в HTML таг.

След идентифициране и успешен влизане, потребителя има достъп до нивата на играта. След натискане на бутона старт се зарежда колелото с въпроси (фиг. 4.10).



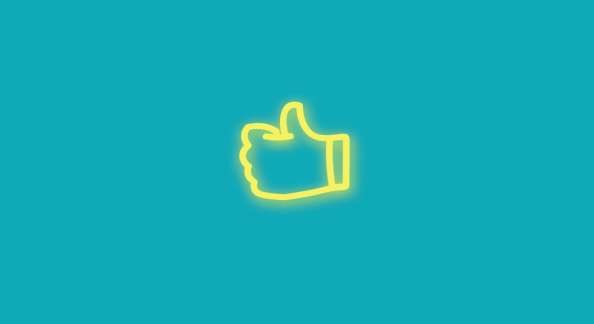


фиг.4.10 Визуализиране на прозрец с въпроси

При натискан на бутона “SPIN” се зарежда на случаен принцип въпрос от базата данни с опции за отговор (фиг. 4.11). Нивото не е за време, играчите имат възможността спокойно да помислят за правилния отговор.

фиг.4.11 Изглед за избор на отговор

При избиране на един от посочените три отговора се появава нов изглед, от който първоначално потребителя разбира дали е познал или не. На фиг. 4.12. и фиг. 4.13 са дадени различните състояния.



фиг.4.12 Изглед за верен отговор

фиг.4.12 Изглед за грешен отговор

Ако играчът реши да прекъсне вече заредения въпрос като натисне директно бутона “OK” се визуализира друг изглед със съобщение за грешка (фиг. 4.13).



фиг.4.13 Изглед за съобщение за грешка

За всеки познат отговор, се добавят +100 точки към общия брой. Когато се зареди и се отговори на последният въпрос, се зарежда изглед с общия брой точки от това ниво, както и линк към следващото (фиг 4.14.).

фиг.4.13 Изглед с общ брой точки

До второ ниво се достига само когато е завършено първо ниво. Зареждат се два реда с маджонг карти (фиг.4.14).



фиг.4.14 Изглед с маджонг карти

Първият ред се състои от имената на различните видове езици за програмиране, предварително представени в началната страница на играта. Вторият ред съдържа фрагмент от код, който е специфичен и уникален за всеки един език.

При натискане на някоя от заредените карти се стартира таймер. В това ниво не се добавят точки към общо събрания брой, а се играе за време. Целта е играча в най-кратък срок да разпознае двойките карти за да приключи нивото (фиг. 4.15, фиг. 4.16).



фиг.4.15



 фиг.4.16

При финализиране на второто ниво, отново се зарежда прозорец със статистика за играта от двете нива (фиг. 4.17).

фиг.4.17 Изглед за статистика

**4.3 Съхранение на резултата**

Резултата от нивата не се съхраняват в базата данни, а чрез *local Storage.* Механизмът за местно съхранение е добър заместител на бисквитките, и с HTML5, може да се запише до 5MB данни в специален файл на клиента. Този файл не е изпълним и

не може да държи двоични данни (само стрингове), така че е безопаснен. Всички страници, които идват от домейна споделят едно и също място. Използването на този механизъм позволява запазването на данни преминаващи между различни страници. Локалното хранилище съхранява данните, като всяка част от тях се състои от двойка ключ/стойност. Първият параметър идентифицира името на информацията, а втория стойността, свързана с този ключ. Съдържат се малък брой методи, които са изключително мощни и лесни за използване:

* localStorage.setItem (ключ, стойност)
* localStorage.getItem (ключ)
* localStorage.removeItem (ключ)
* localStorage.length

Функцията съхраняваща точките от първо ниво е следната:

*var score = parseInt($('#score').html()); score += 100;*

*localStorage.setItem('score', score); if( localStorage.getItem('score') !== null) { var points = localStorage.getItem('score');*

*$('#score').html(points);*

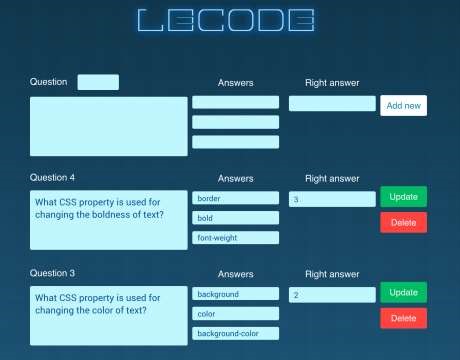
*}*

Избран е механизм за локално съхранение с цел, когато играчите се логнат повторно в играта да имат възможността да играят отново и да постигнат по-добри резултати.

**4.4 Администраторски панел**

Достъп до администраторския панел има само администратора. В таблицата за съхраняване на потребители има колона "admin", в която се указва дали съответният потребител е с администраторски права или не.

Влизането в панела става чрез формата за логин на играчите. Освен достъп до нивата, администратора има линк към още един изглед, в който се съдържат всички въпроси от първо ниво, извлечени от базата данни (фиг.4.18).



фиг.4.18 Изглед за въпроси

Администраторът има възможността да редактира, изтрива или добавя нови въпроси в базата, които ще се отразяват на колелото от първо ниво. Която и от опциите да направи, при правилно въведени данни се появява прозорец с потвърждаващо съобщение (фиг 4.19) последвано от презареждане на страницата.

фиг.4.19 Изглед за потвърждаващо съобщение

*4.5 Колелото на знанията*

За разработката на колелото е използвана JavaScript библиотеката D3.js. Тя предоставя динамични и интерактивни визуализации на обектите. Вградена е в рамките на HTML страницата, библиотеката използва предварително изградени функции на JavaScript, за да избере даден елемент. Създават се SVG обекти, с възможност за добавяне на динамични ефекти или пояснения към тях. Тези обекти могат да бъдат "стилнати" чрез използване на CSS. Силата на D3.js е в генерирането на богати текстови

/ графични схеми и диаграми. Данните могат да бъдат в различни формати като JSON, CSV или GeoJSON както и JavaScript функции за четене на други формати.

Основният принцип на D3.js конструкцията е възможността първо да се използва CSS селектор, който да избере набор от обекти в DOM дървото и с помощта на операторите да се манипулират по начин подобен на jQuery функционалността.

Създаване на колелото:

*var resp = JSON.parse(d.response); data = resp;*

*//създаване на рамери и стил на колелото var padding = {*

*top: 20,*

*right: 40, bottom: 0, left:0*

*},*

*w = 500 - padding.left - padding.right, h = 500 - padding.top - padding.bottom, r = Math.min(w, h)/2, rotation = 0, oldrotation= 0, picked = 100000, oldpick = [], color = d3.scale.category20b();//category20c() различни цветове*

*//randomNumbers = getRandomNumbers(); var spinning = false;*

*//селектиране на eлемент с id = chart и добавяне на svg обект в него var svg = d3.select('#chart')*

*.append("svg")*

*.data([data])*

*.attr("width", w + padding.left + padding.right)*

*.attr("height", h + padding.top + padding.bottom);*

*//добавяне на holder елемет в svg, който ще държи отделните сектори на колелото var container = svg.append("g") .attr("class", "chartholder")*

*.attr("transform", "translate(" + (w/2 + padding.left) + "," + (h/2 + padding.top) + ")");*

*var vis = container.append("g");*

*var pie = d3.layout.pie().sort(null).value(function(d){return 1;});*

*// генериране на фунция за дъга - arc generator function var arc = d3.svg.arc().outerRadius(r);*

*// селектиране на пътя който ще използва дъгата за да се начертае var arcs = vis.selectAll("g.slice")*

*.data(pie)*

*.enter()*

*.append("g")*

*.attr("class", "slice");*

*arcs.append("path")*

*.attr("fill", function(d, i){ return color(i); })*

*.attr("d", function (d) { return arc(d); });*

*// добавяне на текс във всеки един сектор и начина му на завъртане arcs.append("text").attr("transform", function(d){*

*d.innerRadius = 0;*

*d.outerRadius = r;*

*d.angle = (d.startAngle + d.endAngle)/2; return "rotate(" + (d.angle \* 180 / Math.PI - 90) + ")translate(" + (d.outerRadius -10) +")";*

*})*

*.attr("text-anchor", "end").text( function(d, i) { return data[i].label;*

*});*

*container.on("click", spin);*

*//случайно генериране на въпроси function getRandomNumbers(){ //16-bit unsigned integers var array = new Uint16Array(1000);*

*var scale = d3.scale.linear().range([360, 1440]).domain([0, 100000]);*

*if(window.hasOwnProperty("crypto") && typeof window.crypto.getRandomValues*

*=== "function"){*

*window.crypto.getRandomValues(array);*

*} else {*

*//no support for crypto, get crappy random numbers*

*for(var i=0; i < 1000; i++){ array[i] = Math.floor(Math.random() \* 100000) + 1;*

*}*

*}*

*return array;*

*}*

*//завъртане на колелото function rotTween(to) { var i = d3.interpolate(oldrotation % 360, rotation); return function(t) { return "rotate(" + i(t) + ")";*

*};*

*}*

**4.6 Маджонг карти**

Маджонг картите са разработени на JavaScript. Въпросите и отговорите са JSON array, които се викат във функция която всеки път ги позиционира на различни места.

*var cardsQuestions = [*

*{'php':'php'},*

*{'css':'css'},*

*{'ajax':'ajax'},*

*{'html':'html'},*

*{'c++':'c++'},*

*{'javascript':'javascript'},*

*{'java':'java'},*

*];*

*var cardsAnswers = [*

*{'php':'echo "Hello World";'},*

*{'css':' .class{ background: transparent; }'},*

*{'ajax':'$.ajax({ method: "POST", url: "some.php", data: { name: "John", location: "Boston" } })'},*

*{'html':'<aside> Hello World! </aside>'},*

*{'c++':'void main(){int i; return: 0;}'},*

*{'javascript':'console.log("Hello World");'},*

*{'java':'Thread(Runnable threadObj, String threadName);'}*

*];*

*// Shufffel function function shuffle(o){ for(var j, x, i = o.length; i; j = Math.floor(Math.random() \* i), x = o[--i], o[i] = o[j], o[j]*

*= x); return o;*

*}*

*shuffle(cardsQuestions); shuffle(cardsAnswers);*

В цикъл for спрямо дъжината на *cardsQuestions* и *cardsAnswers* се създава вътрешен елемент за всяка карта, съдържащ ключа и стойноста на всеки един от елементите.

*//цикъл for за въпросите for (var i = 0; i < cardsQuestions.length; i++) { var cardQ = document.createElement('div'); var key = Object.keys(cardsQuestions[i])[0]; cardQ.dataset.item = key; cardQ.dataset.view = 'card'; cardQ.innerHTML = '<xmp>'+cardsQuestions[i][key]+'</xmp>';*

*myCardsQ.appendChild(cardQ);*

*}*

*//цикъл for за отговорите*

*for (var i = 0; i < cardsAnswers.length; i++) { var cardA = document.createElement('div'); var key = Object.keys(cardsAnswers[i])[0]; cardA.dataset.item = key; cardA.dataset.view = 'card'; cardA.innerHTML = '<xmp>'+cardsAnswers[i][key]+'</xmp>';*

*myCardsA.appendChild(cardA);*

*}*

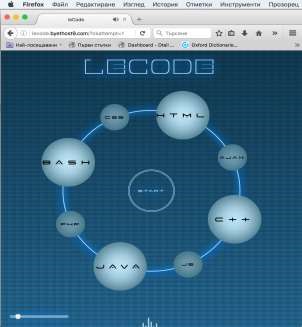
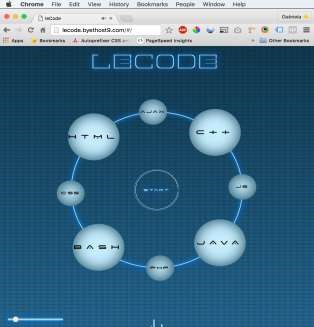
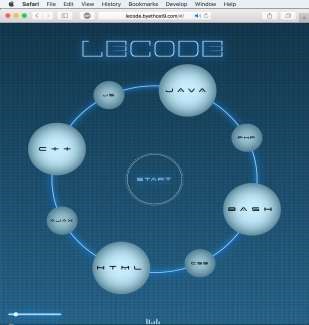
Използва се таг *<xmp></xmp>* за holder (държач) на въпросите и отговорите, като изолира реалния код от отговорите и го визуализара в браузъра като стринг.

**Тестване и настройки**

1. **1 Тестване**

Уеб базираната игра LECODE е проектирана за различни видове браузъри.

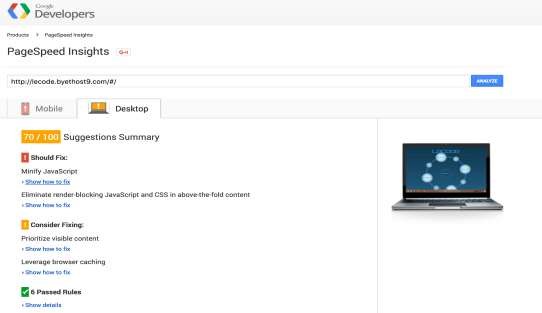
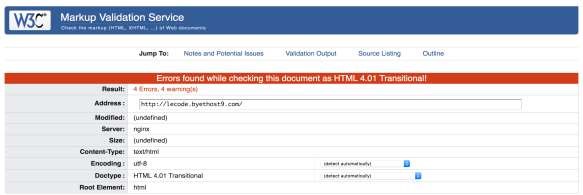
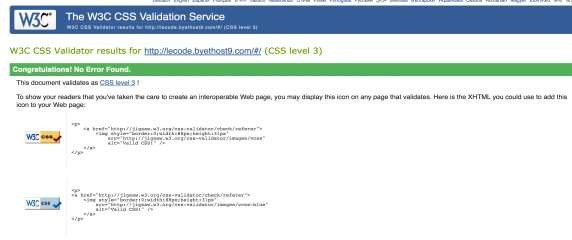
Извършени са тестове на Chrome v.51.0.2704, Firefox v.47.0.1 и Safari v.9.1.1 като е направена допълнителна оптимизиция на част от елементите (фиг.5.1).



фиг.5.1

Извършени са тестове и чрез онлай валидация на HTML Markup и CSS, както и PageSpeed (фиг.5.2)

).



фиг.5.2

**Глава 6 Заключение, анализи, възможности за бъдещо развитие**

1. **1 Заключение**

Проектирането на игри е труден и дълъг процес. Преди да се напише каквато и да е игра, е необходима идея, история, сценарий. Трябва да се определят целите на играта, какъв ще е таргета ѝ, какво ще се включи в нея, за какво ще бъде предназначена и т.н. Добра практика при изграждането на скелета на играта е задаването на въпроси от вида:

* + какво ще прави играча;
  + каква ще бъде целта му;
  + ще има ли определена роля;
  + ще има ли нива и/или поднива;
  + срещу кого ще играе - друг играч, компютър, самостоятелно;
  + какво ще печели - точки, животи, време и др.

Важно е да се наблегне и на потребителския интерфейс. Той трябва да е съобразен с темата на играта и да същевременно да бъде интересен за играчите. Добра практика е честото обновяване на играта, като добавяне на нови нива, увеличаване на степента на трудност, обновяване на визията и др.

**6.2 Възможности за бъдещо развитие**

Разработента игра може да се развие в няколко насоки:

* + увеличаване на броя на възможните въпроси;
  + разширяване на целевата възрастова група;
  + добавяне на различни теми;
  + подобряване на анимациите, част от интерфейса, аудио- и визуалните ефекти
  + оптимизиране на администраторския панел
  + добавяне на всички нивата в админ панела
  + съхранение на резултатите в базата
  + добавяне на респонсив дизайн
  + печелене на титли

Другите възможности за разработване на уеб базирана логическа игра с различни приложения:

* + логически игри за различни възрастови групи; ;
  + игри за професионално обучение на възрастни и учене през целия живот (life-long learning).

**Използвана литература**

1. Даниела Дурева - Тупарова”, Мая Касева”, VII СОУ *“Компютърните образователно игри в обучението в начално училище”* A history of the computer game - [https://www.jesperjuul.net/thesis/2historyofthecomputergame.html](https://www.jesperjuul.net/thesis/2-historyofthecomputergame.html)

1. A history of the computer game -<https://www.jesperjuul.net/thesis/2-historyofthecomputergame.html>

1. Екатерина Бонева-Дамянова”, Николай Цанков, Иво Дамянов, от списание “Образование” бр.4/2012 - “*Компютърните игри – от образователно средство към образователен продукт*

*(за повишаване на мотивацията и интереса)” -*<http://www.obrazovaniebg.net/computer-games/>

1. Software University - *“5 от най-популярните игри, създадени с HTML5” -* <https://judge.softuni.bg/News/Selected/108>

1. HTML5 и Local Storage -<http://dummiesis.com/bg/pages/1484686>

1. phpMyAdmin -<https://www.phpmyadmin.net/>

1. Тодор Тодоров - “*Как да работим с PHP-PDO (Част 1)*” -<https://www.todortodorov.org/blog/94/>

1. D3.js -<https://d3js.org/>

1. PHP -<http://php.net/manual/en/intro.pdo.php>

1. State Manager -<https://github.com/angular-ui/ui-router/wiki>

1. HTML5 Canvas -<https://www.sitepoint.com/html5-canvas-tutorial-introduction/>

1. Egor Kuryanovich, Shy Shalom, Russell Goldenberg, Mathias Paumgarten,David Strauss, Seb LeeDelisle, Gaetan Renaudeau, Jonas Wagner, Jonathan Bergknoff, Rob Hawkes and Brian Danchilla -

“HTML5 Games Most Wanted. Build the Best HTML5 Games” - 2012 -

<http://dominictran.com/pdf/FriendsofED.HTML5.Games.Most.Wanted.Mar.2012.pdf>

1. Wikipedia- History of video i computer games

1. Wikipedia Компютърна игра

1. Wikipedia HTML 5

1. Wikipedia AngularJS

**Приложение**

**index.php**

<!DOCTYPE html>

<html lang="en" ng-app="LeCode">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta http-equiv="PRAGMA" content="NO-CACHE">

<meta name="viewport" id="viewport" content="width=device-width,minimumscale=1.0,maximum-scale=1.0,initial-scale=1.0" />

<meta name="mobile-web-app-capable" content="yes" />

<meta name="apple-mobile-web-app-capable" content="yes" />

<meta name="apple-touch-fullscreen" content="yes">

<title>leCode</title>

<!-- CSS -->

<link href="assets/front/css/style.css" rel="stylesheet">

<!-- Font -->

<link

href='http://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic,400,300,600,800,700' rel='stylesheet' type='text/css'>

</head>

<body ng-cloak="" data-href="<?php echo $\_SERVER['HTTP\_HOST'];?>" ngcontroller="MainController">

<div class="sticky-header" sticky-header></div>

<div class="user-menu-admin"></div>

<div class="user-menu-exit">

<a ng-click="logout()" class="exit"></a>

</div>

<div class="content" ng-class="animation">

<p class="empty-err">All fields are requared.</p>

<p class="data-err">User doesn't exist.</p>

<p class="email-err">The email is incorrect or already use.</p>

<!-- UI Route -->

<div ui-view class="height"></div>

<!-- End UI Route -->

</div>

<footer>

<div ng-controller="AudioController">

<input class="form-control" type=range min=0 max=1 step=0.01 ngmodel="audio.volume">

<div class="bar-c" ng-click="audio.paused ? audio.play() : audio.pause()" ngmodel="audio.loop">

<div id="bar-1" class="bar"></div>

<div id="bar-2" class="bar"></div>

<div id="bar-3" class="bar"></div>

<div id="bar-4" class="bar"></div>

<div id="bar-5" class="bar"></div>

<div id="bar-6" class="bar"></div>

</div>

</div>

</footer>

<script src="//code.jquery.com/jquery-2.1.3.min.js"></script>

<script src="https://d3js.org/d3.v3.min.js" charset="utf-8"></script>

<script src="assets/front/lib/angular.js"></script>

<script src="http://angular-ui.github.io/ui-router/release/angular-ui-router.js"></script>

<script src="assets/front/lib/angular-route.js"></script>

<script src="assets/front/lib/angular-resource.js"></script>

<script src="assets/front/lib/angular-animate.js"></script>

<script src="assets/front/lib/angular-touch.js"></script>

<script src="assets/front/lib/angular.audio.js"></script>

<script src="assets/front/js/scripts.js"></script>

</body>

</html>

home.html

<div class="page page--home active" id="home">

<div class="wheel">

<span class="circles circles--php">BASH</span>

<span class="circles circles--css small">CSS</span>

<span class="circles circles--javascript">HTML</span>

<span class="circles circles--ajax small">AJAX</span>

<span class="circles circles--cplus">C++</span>

<span class="circles circles--html small">JS</span>

<span class="circles circles--java">JAVA</span>

<span class="circles circles--bash small">PHP</span>

<!-- <a href="#login" class="for-login main-circle">Start</a> -->

<a href="#app" class="logged main-circle">Start</a>

</div>

</div>

application.html

<div class="wrap-wheel flex flex--column">

<div id="score">0</div>

<div id="chart"></div>

<div id="question" class="flex flex--column">

<h1></h1>

</div>

</div>

<div class="message empty-option flex flex--column">

<p>Please select one of the options.</p>

<span class="btn-ok">OK</span>

</div>

<div class="message success">

<img src="assets/front/images/like.png" alt="">

</div>

<div class="message error">

<img src="assets/front/images/unlike.png" alt="">

</div>

<script type="text/javascript" charset="utf-8">

if( localStorage.getItem('challenge1') !== null) {

location.href= 'http://' + $('body').data('href') + '#statistics';

}

d3.xhr("endpoints/questions.php").header("Content-Type",

"application/json").post(function(error, d) {

if( localStorage.getItem('score') !== null) { var points = localStorage.getItem('score');

$('#score').html(points);

}

var resp = JSON.parse(d.response);

data = resp; var padding = {

top: 20, right: 40, bottom: 0, left:0

},

w = 500 - padding.left - padding.right,

h = 500 - padding.top - padding.bottom, r = Math.min(w, h)/2, rotation = 0, oldrotation= 0, picked = 100000,

oldpick = [],

color = d3.scale.category20b();//category20c() //randomNumbers = getRandomNumbers();

var spinning = false;

var svg = d3.select('#chart')

.append("svg")

.data([data])

.attr("width", w + padding.left + padding.right)

.attr("height", h + padding.top + padding.bottom);

var container = svg.append("g") .attr("class", "chartholder") .attr("transform", "translate(" + (w/2 + padding.left) + "," + (h/2 + padding.top) +

")");

var vis = container.append("g");

var pie = d3.layout.pie().sort(null).value(function(d){return 1;});

// declare an arc generator function

var arc = d3.svg.arc().outerRadius(r);

// select paths, use arc generator to draw

var arcs = vis.selectAll("g.slice")

.data(pie)

.enter()

.append("g")

.attr("class", "slice");

arcs.append("path")

.attr("fill", function(d, i){ return color(i); })

.attr("d", function (d) { return arc(d); });

// add the text

arcs.append("text").attr("transform", function(d){

d.innerRadius = 0;

d.outerRadius = r;

d.angle = (d.startAngle + d.endAngle)/2;

return "rotate(" + (d.angle \* 180 / Math.PI - 90) + ")translate(" + (d.outerRadius -

10) +")";

})

.attr("text-anchor", "end") .text( function(d, i) {

return data[i].label;

});

container.on("click", spin);

var answeredQuestions = [];

if( localStorage.getItem("answeredQuestions")!== null) { var storedansweredQuestions =

JSON.parse(localStorage.getItem("answeredQuestions"));

oldpick = storedansweredQuestions;

for (var i = 0; i < oldpick.length; i++) {

d3.select(".slice:nth-child(" + (oldpick[i] + 1) + ") path").attr("fill", "#dcdcdc");

}

}

function spin(d){

|  |  |  |
| --- | --- | --- |
|  |  | // if( spinning === true ) |
|  |  | // return; |
|  |  | // spinning = true; |
|  |  | //all slices have been seen, all done |
|  |  | // if(oldpick.length == (data.length)){ |
|  |  | // // $() |
|  |  | // container.on("click", null); |
|  |  | // return; |
|  |  | // } |
|  |  | var ps = 360/data.length, |
|  |  | pieslice = Math.round(1440/data.length), |
|  |  | rng = Math.floor((Math.random() \* 1440) + 360); |
|  |  |  |
|  |  | rotation = (Math.round(rng / ps) \* ps); |
|  |  |  |
|  |  | picked = Math.round(data.length - (rotation % 360)/ps); |
|  |  | picked = picked >= data.length ? (picked % data.length) : picked; |
|  |  | if(oldpick.indexOf(picked) !== -1){ |
|  |  | d3.select(this).call(spin); |
|  |  | return; |
|  |  | } else { |
|  |  | oldpick.push(picked); |
|  |  | localStorage.setItem("answeredQuestions", JSON.stringify(oldpick)); |
|  |  | } |
|  |  | //stoping rotate to selected pie |
|  |  | rotation += 90 - Math.round(ps/2); |
| function(){ |  | vis.transition().duration(3000).attrTween("transform", rotTween).each("end", |
|  |  | var score = parseInt($('#score').html()); |
|  |  | score +=100; |
|  |  | //mark question as seen |
| "#dcdcdc"); |  | d3.select(".slice:nth-child(" + (picked + 1) + ") path").attr("fill", |
| checked"); |  | d3.select(".slice:nth-child(" + (picked + 1) + ")").attr("class", "slice |
|  |  | //populate question |
|  |  | var rpcUrl = "endpoints/post.php"; |
|  |  | var rpcData = '{"id": "' + (picked + 1) + '" }'; |
|  |  | d3.xhr(rpcUrl).header("Content-Type", "application/json").post(rpcData, |

function(error, data) {

var resp = JSON.parse(data.response);

var ans = JSON.parse(resp.answers);

$("#question").append("<h1>"+resp.question+"</h1>");

var html = '<form name="answers">'; html += '<input type="hidden" name="question\_id"

id="question\_id" value="' + (picked + 1) + '">';

$(ans).each( function(element, index) {

var \_this = this;

html += "<label><input type='radio' name='answer[]'

value='"+this.id+"'>" + \_this.answer + "<span></span></label>";

});

html +='<input type="submit" value="OK" id="send-ans">';

html +='</form>';

$('#question').append(html);

$('#question').addClass('active');

});

// setTimeout(function() {

// spinning = false;

// }, 1000);

oldrotation = rotation;

});

}

//make arrow

svg.append("g")

.attr("transform", "translate(" + (w + padding.left + padding.right) + "," + ((h/2)+padding.top) + ")")

.append("path")

.attr("d", "M-" + (r\*.15) + ",0L0," + (r\*.05) + "L0,-" + (r\*.05) + "Z")

.style({"fill":"black"});

//draw spin circle container.append("circle")

.attr("cx", 0)

.attr("cy", 0)

.attr("r", 60)

.style({"fill":"#ffffff","cursor":"pointer"});

//spin text

container.append("text")

.attr("x", 0)

.attr("y", 15)

.attr("text-anchor", "middle")

.text("SPIN")

.style({"font-weight":"bold", "font-size":"30px","cursor":"pointer"});

function rotTween(to) {

var i = d3.interpolate(oldrotation % 360, rotation);

return function(t) {

return "rotate(" + i(t) + ")";

};

}

function getRandomNumbers(){

var array = new Uint16Array(1000);

var scale = d3.scale.linear().range([360, 1440]).domain([0, 100000]);

if(window.hasOwnProperty("crypto") && typeof

window.crypto.getRandomValues === "function"){

window.crypto.getRandomValues(array);

} else {

//no support for crypto, get crappy random numbers for(var i=0; i < 1000; i++){

array[i] = Math.floor(Math.random() \* 100000) + 1;

}

}

return array;

}

});

</script>

cards.html

<div class="page page--cards active" id="cards">

<div class="wrapper">

<h3 class="style">Click any card to begin</h3>

<div class="time">

<span id="seconds">00</span>:<span id="tens">00</span>

</div>

<p id ="text"></p>

<label for="">Languages</label>

<div id="container-q" class="container">

</div>

<label for="">Code snippet</label>

<div id="container-a" class="container">

</div>

</div>

</div>

<script>

var myCardsQ = document.getElementById('container-q'); var myCardsA = document.getElementById('container-a'); var resultsArray = [];

var counter = 0;

var text = document.getElementById('text'); var seconds = 0; var tens = 0;

var appendTens = document.getElementById('tens');

var appendSeconds = document.getElementById('seconds');

var Interval ;

var cardsQuestions = [

{'php':'php'},

{'css':'css'},

{'ajax':'ajax'},

{'html':'html'},

{'c++':'c++'},

{'javascript':'javascript'},

{'java':'java'},

];

var cardsAnswers = [

{'php':'echo "Hello World";'},

{'css':' .class{ background: transparent; }'},

{'ajax':'$.ajax({ method: "POST", url: "some.php", data: { name: "John", location:

"Boston" } })'},

{'html':'<aside> Hello World! </aside>'},

{'c++':'void main(){int i; return: 0;}'},

{'javascript':'console.log("Hello World");'},

{'java':'Thread(Runnable threadObj, String threadName);'}

];

var clicable = true;

var flippedArray = [];

// Shufffel function function shuffle(o){

for(var j, x, i = o.length; i; j = Math.floor(Math.random() \* i), x = o[--i], o[i] = o[j], o[j]

= x);

return o;

}

shuffle(cardsQuestions); shuffle(cardsAnswers);

// var card;

for (var i = 0; i < cardsQuestions.length; i++) {

var cardQ = document.createElement('div'); var key = Object.keys(cardsQuestions[i])[0];

cardQ.dataset.item = key; cardQ.dataset.view = 'card';

cardQ.innerHTML = '<xmp>'+cardsQuestions[i][key]+'</xmp>'; myCardsQ.appendChild(cardQ);

}

for (var i = 0; i < cardsAnswers.length; i++) {

var cardA = document.createElement('div'); var key = Object.keys(cardsAnswers[i])[0];

cardA.dataset.item = key; cardA.dataset.view = 'card';

cardA.innerHTML = '<xmp>'+cardsAnswers[i][key]+'</xmp>';

myCardsA.appendChild(cardA);

}

$(document).on('click', 'div[data-view="card"]', function () { if(clicable == false)

return;

if ( !$(this).hasClass('flipped') && !$(this).hasClass('correct') ){

var result = $(this).data('item');

var \_this = $(this);

$(this).attr('class', '').addClass('flipped'); resultsArray.push($(this)); clearInterval(Interval);

Interval = setInterval(startTimer, 10);

}

if (resultsArray.length == 2 ) {

if ($(resultsArray[0]).data('item') === $(resultsArray[1]).data('item') ) {

check('correct'); counter ++; win();

resultsArray = [];

}

} else if (resultsArray.length > 2) {

flippedArray = resultsArray;

$(flippedArray[0]).attr('class', '').addClass('reverse'); $(flippedArray[1]).attr('class', '').addClass('reverse');

console.log(flippedArray[0], flippedArray[1]);

resultsArray.shift(); resultsArray.shift();

}

});

var check = function(className) { var x = document.getElementsByClassName('flipped'); setTimeout(function() {

for(var i = (x.length - 1); i >= 0; i--) {

x[i].className = className;

}

},500);

};

var win = function () {

if(counter === (cardsQuestions.length) && counter === (cardsAnswers.length) ) {

clearInterval(Interval);

localStorage.setItem('challenge2', 'Your time was ' + seconds + ':' + tens); location.href= 'http://' + $('body').data('href') + '#statistics';

}

};

function startTimer () {

tens++;

if(tens < 9){

appendTens.innerHTML = '0' + tens;

}

if (tens > 9){

appendTens.innerHTML = tens;

}

if (tens > 99) {

seconds++;

appendSeconds.innerHTML = '0' + seconds; tens = 0;

appendTens.innerHTML = '0' + 0;

}

if (seconds > 9){ appendSeconds.innerHTML = seconds;

}

}

</script>

**login.html**

<div class="form flex flex--column" id="login">

<h2>Login</h2>

<input type="text" class="form-control" ng-model="loginInfo.email" name="email" placeholder="Email or username">

<input type="password" class="form-control" ng-model="loginInfo.password" placeholder="Password">

<input type="submit" ng-click="loginUser()" value="Sign In">

<span>or</span>

<a href="#signup">Sign up</a>

</div>

signup.html

<div class="form flex flex--column" id="signup">

<h2>Sign Up</h2>

<!-- <form action="" method="POST"> -->

<input type="text" class="form-control" ng-model="signUpInfo.email" name="email" placeholder="Email or username">

<input type="password" class="form-control" ng-model="signUpInfo.password" name="password" placeholder="Choose Password">

<input type="submit" value="Sign Up" ng-click="signUserUp()" name="signup"/>

<!-- </form> -->

<span>or</span>

<a href="#login">Sign in</a>

</div>

statistics.html

<!-- <a ng-click="logout()" class="exit">logout</a> -->

<div class="statistics flex flex--column">

<p>your score</p>

<div class="total-score score"></div>

<div class="challenge2 score"></div>

<div class="flex level">

<a href="#cards">Next challenge</a>

</div>

</div>

<script>

if( localStorage.getItem('score') !== null) { var points = localStorage.getItem('score');

$('.total-score').html('<p>Challenge 1: <span>'+points+' points</span></p>');

}

if( localStorage.getItem('challenge2') !== null) { var time = localStorage.getItem('challenge2');

$('.challenge2').html('<p>Challenge 2: <span>'+time+'</span></p>'); $('.level').hide();

}

</script>

checkToken.php

<?php

include ('../connection.php');

$data = json\_decode(file\_get\_contents("php://input"));

$token = $data->token;

$check = $db->prepare("SELECT \* FROM users WHERE token=:token");

$check->execute(array(

":token" => $token,

));

$check->fetch();

if (!empty($check)) {

echo "authorized";

} else {

echo "unauthorized";

}

?>

login.php

<?php

include ('../connection.php');

$data = json\_decode(file\_get\_contents("php://input"));

$email = $data->email;

$password = sha1($data->password);

$userInfo = $db->prepare("SELECT \* FROM users WHERE email=:email AND password=:password"); $userInfo->execute(array(

":email" => $email,

":password" => $password,

));

$result = $userInfo->fetch();

if (!empty($result)) {

//When user is logged in and let's givem a token

$token = $email." | ".uniqid().uniqid().uniqid();

$q = $db->prepare("UPDATE users SET token=:token WHERE id=:id");

$q->execute(array(

":token" => $token,

":id" => $result['id'],

));

echo json\_encode(array('user\_id' => $result['id'], 'admin' => $result['admin'], 'token' => $token));

} else {

echo 'ERROR';

}

?>

logout.php

<?php

include ('../connection.php');

$data = json\_decode(file\_get\_contents("php://input"));

$token = $data->token;

$logout = $db->prepare("UPDATE users SET token='' WHERE token=:token");

$logout->execute(array(

":token" => $token,

));

?>

post.php

<?php

include ('../connection.php');

$foo = file\_get\_contents('php://input');

$array = json\_decode($foo);

$data = $db->prepare('SELECT \* FROM questions WHERE id\_question=:id');

$data->execute(array(

":id" => $array->id,

));

$newRow = $data->fetch(); // print\_r($newRow);

echo json\_encode(array('question' => $newRow['question'], 'answers' => $newRow['answers'])); ?>

question-edit.php

<?php

include ('../connection.php');

if (isset($\_POST['action']) && $\_POST['action'] == 'add') {

$arr = [];

foreach ($\_POST['answers'] as $key => $value) {

$arr[$key]['id'] = $key+1;

$arr[$key]['answer'] = $value;

}

$dataAdd = $db->prepare('INSERT INTO questions (id\_question, question, answer, answers) VALUES (:id\_question, :question, :answer, :answers)');

$dataAdd->execute(array(

":id\_question" => $\_POST['id\_question'],

":question" => $\_POST['question'],

":answer" => $\_POST['answer'],

":answers" => json\_encode($arr),

));

echo 'add';

}

//Update question in database

if (isset($\_POST['action']) && $\_POST['action'] == 'update') {

$arr = [];

foreach ($\_POST['answers'] as $key => $value) {

$arr[$key]['id'] = $key+1;

$arr[$key]['answer'] = $value;

}

$dataUpdate = $db->prepare('UPDATE questions SET question=:question, answers=:answers, answer=:answer WHERE id\_question=:id\_question');

$dataUpdate->execute(array(

":id\_question" => $\_POST['id\_question'],

":question" => $\_POST['question'],

":answers" => json\_encode($arr),

":answer" => $\_POST['answer'],

));

echo 'update';

}

//Delete question in database

if (isset($\_POST['action']) && $\_POST['action'] == 'delete') {

$dataDelete = $db->prepare('DELETE FROM questions WHERE id\_question=:id\_question');

$dataDelete->execute(array(

":id\_question" => $\_POST['id\_question'],

));

echo 'delete';

}

?>

questions.php <?php

include ('../connection.php');

$data = $db->query('SELECT \* FROM questions');

$newRow = $data->fetchAll();

$arr = [];

foreach ($newRow as $key => $value) {

$arr[$key]['label'] = 'Question '.($key+1);

$arr[$key]['question\_id'] = $value['id\_question'];

}

echo json\_encode($arr);

?>

**signup.php**

<?php

include ('../connection.php');

$data = json\_decode(file\_get\_contents("php://input"));

$email = $data->email;

$password = sha1($data->password);

$emailexist = $db->prepare("SELECT \* FROM users WHERE email=:email");

$emailexist->execute(array(

":email" => $email,

));

$res = $emailexist->fetch(); if (empty($res)) {

$q = $db->prepare("INSERT INTO users (email, password) VALUES (:email, :password)");

$q->execute(array(

":email" => $email,

":password" => $password,

));

if ($q) {

echo 'OK';

} else { echo 'ERROR';

}

} else {

echo 'Email exist!';

}

?>

validation.php

<?php

include ('../connection.php');

$question\_id = $\_POST['questionId'];

$answer\_id = $\_POST['answerId'];

$user\_id = $\_POST['userId'];

$data = $db->prepare('SELECT \* FROM questions WHERE id\_question=:id'); $data->execute(array(

":id" => $question\_id,

));

$newRow = $data->fetch();

if ($answer\_id == $newRow['answer']) {

echo 'success';

} else {

echo 'error';

}

//insert in database

// $insert = $db->prepare('INSERT INTO user\_answers VALUES("", :user\_id, :question\_id, :answer\_id)');

// $insert->execute(array(

// ":user\_id" => $user\_id,

// ":question\_id" => $question\_id,

// ":answer\_id" => $answer\_id,

// ));

?>

app.js

'use strict';

var app = angular.module('LeCode', ['ngRoute', 'ngResource', 'ngAnimate', 'ngTouch', 'ngAudio', 'ui.router']);

app.config(function($stateProvider, $urlRouterProvider){

$urlRouterProvider.otherwise('/');

$stateProvider

.state('login', {

url:'/login',

controller: 'LoginController', templateUrl: 'views/login.html',

animation: 'animate-page'

})

.state('signup', {

url:'/signup',

controller: 'LoginController', templateUrl: 'views/signup.html',

animation: 'animate-page'

})

.state('application', {

url:'/app',

controller: 'MainController', templateUrl: 'views/application.html', animation: 'animate-page'

// params: {'test': null}

})

.state('cards', {

url:'/cards',

controller: 'MainController', templateUrl: 'views/cards.html', animation: 'animate-page'

// params: {'test': null}

})

.state('statistics', {

url:'/statistics', controller: 'MainController',

templateUrl: 'views/statistics.html', animation: 'animate-page'

// params: {'test': null}

})

.state('admin', {

url:'/admin',

controller: 'MainController', templateUrl: 'views/admin/admin.php', animation: 'animate-page'

// params: {'test': null}

})

.state('index', {

url:'/',

templateUrl: 'views/home.html',

animation: 'animate-page'

})

});

app.controller('AudioController', function( $scope, ngAudio ) {

$scope.audio = ngAudio.play('assets/front/audio/This-Sky-of-Mine.mp3'); // returns NgAudioObject

$scope.audio.volume = 0.1;

$scope.audio.loop = true;

$scope.clicked = true;

if( $scope.audio.volume === 0 ) { $scope.audio = ngAudio.stop();

angular.element($('.bar').addClass('noAnim'));

} else {

$('.bar').removeClass('noAnim');

}

});

app.directive('stickyHeader', function($window, $document) { return {

restrict: 'EA',

replace: true, scope: { cssClassHidden: '@?',

showAtBottom: '@?'

},

transclude: true,

templateUrl: 'views/common/header.html'

};

});

//set active navigation links

app.directive('activeLink', function($location) {

return { restrict: 'EA',

scope: {

path: "@activeLink"

},

link: function(scope, element, attributes) {

scope.location = $location;

scope.$on('$locationChangeSuccess', function(el) {

if ($location.path() === scope.path) {

element.addClass('active');

} else {

element.removeClass('active');

}

});

scope.$watchCollection(location, function(currentPath) {

currentPath = $location.path().replace('/', '#');

if( currentPath === element[0].attributes['href'].nodeValue ) { element.addClass('active');

} else {

element.removeClass('active');

}

});

}

};

});

app.service('AuthenticationService', ['$http', '$state', function($http, $state){ var self = this;

self.checkToken = function(token) { var data = {token: token};

$http.post('endpoints/checkToken.php', data).success(function(response){

if (response == 'unauthorized'){

$state.go('login');

$('.user-menu-exit').removeClass('show');

} else {

if(localStorage.getItem('admin') !== null){ $('.user-menu-admin').html('<a href="#admin" class="admin">admin</a>');

}

// $('.user-menu-exit').html('<a ng-click="logout()" class="exit"></a>');

$('.user-menu-exit').addClass('show');

return response;

}

}).error(function(error){

$state.go('login');

$('.user-menu-exit').removeClass('show');

});

}

}]);

app.controller('MainController', function ($scope, $state, $http, AuthenticationService){

//If user is not logged in var token;

if (localStorage['token']) {

token = JSON.parse(localStorage['token']); AuthenticationService.checkToken(token);

} else {

$state.go('login');

}

$scope.logout = function(){

var data = {

token: token

}

$http.post('endpoints/logout.php', data).success(function(response){

// console.log(response) localStorage.clear();

$state.go('login');

$('.user-menu-exit').removeClass('show');

$('.user-menu-admin').html('');

}).error(function(error){

console.error(error);

})

}

});

app.controller('LoginController', function($scope, $http, $state){

//Variables

$scope.signUpInfo = { email: undefined,

password: undefined

}

$scope.loginInfo = {

email: undefined,

password: undefined

}

var result = {

test: undefined

}

result.test = JSON.stringify(result.test);

//Functions

$scope.signUserUp = function () {

var data = {

email: $scope.signUpInfo.email,

password: $scope.signUpInfo.password

}

if(typeof data.email !== 'undefined' && data.password !== 'undefined' ) { $http.post('endpoints/signup.php', data).success(function(response) {

if(response == 'Email exist!') {

// localStorage.setItem('token', JSON.stringify(response));

//error email exit view

$('#signup input:not([type="submit"])').addClass('err');

$('.email-err').addClass('active'); setTimeout(function(){

$('#signup input:not([type="submit"])').removeClass('err');

$('.email-err').removeClass('active');

}, 5000);

} else if(response === 'ERROR'){

console.log('SQL ERROR');

} else {

$state.go('login');

}

}).error(function(error) {

console.error(error);

});

} else {

$state.go('signup');

$('#signup input:not([type="submit"])').addClass('err');

$('.empty-err').addClass('active');

setTimeout(function(){

$('#signup input:not([type="submit"])').removeClass('err');

$('.empty-err').removeClass('active');

}, 5000);

}

};

$scope.loginUser = function () {

var data = {

email: $scope.loginInfo.email,

password: $scope.loginInfo.password

}

if(typeof data.email !== 'undefined' && data.password !== 'undefined' ) {

$http.post('endpoints/login.php', data).success(function(response) {

if(response !== 'ERROR') { $('.logged').addClass('active'); $('.for-login').addClass('hide');

localStorage.setItem('token', JSON.stringify(response.token)); localStorage.setItem('user\_id', response.user\_id); if (response.admin) {

localStorage.setItem('admin', response.admin);

}

$state.go('application', result);

// console.log(result);

} else {

$('#login input:not([type="submit"])').addClass('err');

$('.data-err').addClass('active'); setTimeout(function(){

$('#login input:not([type="submit"])').removeClass('err'); $('.data-err').removeClass('active');

}, 5000);

}

}).error(function(error) {

console.error(error);

});

} else {

//error view 2

$('#login input:not([type="submit"])').addClass('err');

$('.empty-err').addClass('active');

setTimeout(function(){

$('#login input:not([type="submit"])').removeClass('err');

$('.empty-err').removeClass('active');

}, 5000);

}

}

});

scripts.js

/\*global jQuery:false, Browser:false, console: false, alert:false \*/

(function($) {

'use strict';

$(function() {

var send = new SendAnswer();

});

function SendAnswer() {

$(document).on('submit', 'form[name="answers"]', function(e){

e.preventDefault();

var checked = $(this).find('input[type="radio"]:checked'); if( checked.length === 0) {

$('.message.empty-option').addClass('active');

$('.message.empty-option .btn-ok').on( Browser.click(), function() {

$('.message.empty-option').removeClass('active');

});

} else {

var questionId = $('#question\_id').val(); var answerId = checked.val(); var userId = localStorage.getItem('user\_id');

$.ajax({

method: 'POST',

url: 'endpoints/validation.php',

data: {

questionId: questionId, answerId: answerId ,

userId: userId

}

})

.done(function( msg ) { var sliceCount = $('.slice').length; var sliceChecked = $('g.checked').length;

if(msg === 'success') {

var score = parseInt($('#score').html());

score += 100;

localStorage.setItem('score', score);

$('#score').html(score);

$('.message.success').addClass('active');

$('.message.success').on( Browser.click(), function() {

$(this).removeClass('active');

});

setTimeout( function() {

$('.message.success').removeClass('active'); if(sliceCount === sliceChecked) { localStorage.setItem('challenge1', 1); location.href= 'http://' + $('body').data('href')

+ '#statistics';

}

},3000);

} else {

$('.message.error').addClass('active');

$('.message.error').on( Browser.click(), function() {

$(this).removeClass('active');

});

setTimeout( function() {

$('.message.error').removeClass('active'); if(sliceCount === sliceChecked) { localStorage.setItem('challenge1', 1); location.href ='http://' + $('body').data('href')

+ '#statistics';

}

},3000);

}

$('#question').html('');

$('#question').removeClass('active');

});

}

});

}

})(jQuery);

**style.css**

@import

url(http://fonts.googleapis.com/css?family=Roboto:300,400,500,600|Roboto+Condensed:400,700|Quick sand:400);

@import url(https://fonts.googleapis.com/css?family=Seymour+One&subset=latin,cyrillic);

@import url(https://fonts.googleapis.com/css?family=Press+Start+2P&subset=latin,cyrillic); @import "sprite.less";

@import "../../modules/less/reset.less";

@import "../../modules/less/flex.less";

@import "../../modules/less/animation.less";

@import "../../modules/less/toaster.less";

@import "../../modules/less/fonts.less";

html, body {

box-sizing: border-box;

-webkit-overflow-scrolling: touch; -webkit-font-smoothing: antialiased; text-rendering: geometricPrecision; -moz-osx-font-smoothing: grayscale;

-webkit-text-size-adjust: 100%;

-webkit-touch-callout: none; -webkit-backface-visibility: hidden; font-family: "Roboto", sans-serif;

line-height: 20px;

}

html {

font-size: 100%;

height: 100%;

}

html, body, h1, h2, h3, h4, h5, h6, p, a, span { -webkit-font-smoothing: antialiased; text-rendering: optimizeLegibility; -moz-osx-font-smoothing: grayscale;

-webkit-text-size-adjust: 100%;

-webkit-overflow-scrolling: touch;

-webkit-touch-callout: none;

}

a, span, label {

font-family: "Press Start 2P", cursive;

}

input, label, span {

-webkit-tap-highlight-color: rgba(0,0,0,0);

}

input, textarea {

-webkit-appearance: none;

}

body {

position: relative; width: 100%; font-size: 16px; color: #000; font-weight: 400; background: #ffffff; display: flex; // height: 100%; // min-height: 100%; flex-direction: column; // background: #061117;

// /\* Permalink - use to edit and share this gradient: http://colorzilla.com/gradienteditor/#1e5799+0,084469+50,061117+100 \*/ // background: #1e5799; /\* Old browsers \*/

// background: -moz-radial-gradient(center, ellipse cover, #1e5799 0%, #084469 50%, #061117 100%); /\* FF3.6-15 \*/

// background: -webkit-radial-gradient(center, ellipse cover, #1e5799 0%,#084469

50%,#061117 100%); /\* Chrome10-25,Safari5.1-6 \*/

// background: radial-gradient(ellipse at center, #1e5799 0%,#084469 50%,#061117 100%); /\*

W3C, IE10+, FF16+, Chrome26+, Opera12+, Safari7+ \*/

// filter: progid:DXImageTransform.Microsoft.gradient( startColorstr="#1e5799", endColorstr="#061117",GradientType=1 ); /\* IE6-9 fallback on horizontal gradient \*/ background-size: cover;

&:not(.home) {

.intro {

padding-top: 4.375rem;

}

}

&:before { content: '';

position: absolute; width: 100%;

height: 100%;

z-index: -2;

top: 0;

left: 0;

background: linear-gradient(0deg,#1a648c,#0e4669,#084367,#4ebbff,#184965) repeat; background-size: 1000% 800%;

// animation: AnimationName 14s ease infinite;

}

&:after { content: ''; position: absolute; width: 100%; height: 100%; z-index: -1; top: 0; left: 0; right: 0; bottom: 0;

margin: auto;

background:url(data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAQAAAAECA YAAACp8Z5+AAAAGUlEQVQIW2NkgID/DAwMjCAaRKAAmABuFQCcoAQDtXM31gAAAABJ

RU5ErkJggg==) repeat;

}

}

a {

color: inherit;

}

.flex {

display: flex;

&--row { flex-flow: row;

justify-content: space-between;

}

&--column { flex-flow: column;

}

}

.form {

h2 {

margin-bottom: 24px;

#Font > .SpaceAgeCyrillic(); color: #AFD4EC;

text-shadow: 0px 0px 18px #0C89EC, 0 0 0 #2196F3, 0px 2px 25px #0F71D2;

font-size: 25px;

}

input { margin-bottom: 20px;

background: none;

}

input:not([type="submit"]) {

background: none; border: 1px solid #AFD4EC;

box-shadow: 0px 0px 2px #0C89EC, 0 0 0 #2196F3, 0px 2px 25px #0F71D2; border-radius: 3px; height: 50px; line-height: 50px; color: #AFD4EC; padding: 0 20px; width: 300px;

font-size: 17px;

}

input[type="submit"] {

background: none; border: 1px solid #AFD4EC;

box-shadow: 0px 0px 2px #0C89EC, 0 0 0 #2196F3, 0px 2px 25px #0F71D2; border-radius: 3px; height: 50px; line-height: 50px; background: #AFD4EC; width: 300px;

font-size: 17px;

}

a, span { color: #AFD4EC;

text-shadow: 0px 0px 18px #0C89EC, 0 0 0 #2196F3, 0px 2px 25px #0F71D2;

#Font > .SpaceAgeCyrillic();

}

a { color: #AFD4EC;

font-size: 24px;

}

span {

font-size: 16px;

}

}

main {

position: relative;

z-index: 5;

}

.logo, .style {

// width: 100%; // margin-bottom: 30px; text-align: center; #Font > .SpaceNeon(); font-size: 60px; line-height: 100px;

color: #AFD4EC;

// text-shadow: 0px 0 28px rgb(0, 150, 245);

text-shadow: 0px 0px 18px #0C89EC, 0 0 0 #2196F3, 0px 2px 25px #0F71D2; letter-spacing: 7px;

@media only screen and(max-height: 766px) { font-size: 50px;

line-height: 60px;

}

}

.style {

font-size: 25px;

font-family: 'Helvetica',sans-serif;

font-weight: 300;

}

@keyframes AnimationName {

0%{background-position:50% 0%}

50%{background-position:51% 100%}

100%{background-position:50% 0%}

}

// .light {

|  |  |
| --- | --- |
| // | position: absolute; |
| // | width: 100%; |
| // | height: 100%; |
| // | z-index: -1; |
| // | top: 0; |
| // | left: 0; |
| // | right: 0; |
| // | bottom: 0; |
| //  // | margin: auto; |
|  | background:url(data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAQAAAAECA |

YAAACp8Z5+AAAAGUlEQVQIW2NkgID/DAwMjCAaRKAAmABuFQCcoAQDtXM31gAAAABJ

RU5ErkJggg==) repeat;

// // background:url("../images/bg-main.png") no-repeat center;

// // background-size: 1875px 1080px;

// // background: rgb(187, 229, 255);

// // box-shadow: 0 0 227px rgb(173, 221, 255),0 24px 227px #afd4e8,0 0 227px

rgba(109,178,226,.6),59px 64px 227px rgba(147,202,234,.6),-144px 14px 227px 0 rgb(112, 208, 236);

// }

.text{

font-family:Helvetica, Arial, sans-serif;

font-size:11px;

pointer-events:none;

}

// audio style

.bar-c {

overflow: hidden; height: 30px; width: 40px; margin: 0 auto; position: relative;

cursor:pointer;

}

.bar {

position: absolute; height: 1px;

width: 3px;

display: inline-block; background-color: #AFD4EC;

bottom: 0;

&.noAnim { animation: none;

animation-name: none;

}

&#bar-1 {

left: 0;

animation-name: danceHeight1;

animation-delay: 0ms;

}

&#bar-2 {

left: 6px;

animation-name: danceHeight2;

animation-delay: 300ms;

}

&#bar-3 {

left: 12px; animation-name: danceHeight3; animation-delay: 600ms;

}

&#bar-4 {

left: 18px;

animation-name: danceHeight4;

animation-delay: 900ms;

}

&#bar-5 {

left: 24px;

animation-name: danceHeight5;

animation-delay: 1200ms;

}

&#bar-6 {

left: 30px;

animation-name: danceHeight6;

animation-delay: 1500ms;

}

animation-duration: 500ms; animation-play-state: running; animation-direction: alternate; animation-timing-function: linear;

animation-iteration-count: infinite;

}

@keyframes danceHeight1 { from { height: 1px; }

to { height: 22px; }

}

@keyframes danceHeight2 { from { height: 1px; }

to { height: 15px; }

}

@keyframes danceHeight3 { from { height: 1px; }

to { height: 30px; }

}

@keyframes danceHeight4 { from { height: 1px; }

to { height: 20px; }

}

@keyframes danceHeight5 { from { height: 1px; }

to { height: 15px; }

}

@keyframes danceHeight6 { from { height: 1px; }

to { height: 15px; }

}

// end audio style

.btns {

#Sprite > .btns(); width: 300px; height: 75px; line-height: 75px; text-align: center; color: #0096f5; font-size: 25px; justify-content: center; margin-bottom: 30px; transition: all .3s ease;

// @media only screen and (min-width: 1025px) {

&:hover {

#Sprite > .btns(#f0cf62);

color: #f0cf62;

}

// }

}

.animate-page .page.active { position: absolute; top: 0;

left: 0; height: 100vh; background-color: #fff;

}

//header

.sticky-header { position: relative; z-index: 5; height: 80px; padding-top: 20px; justify-content: center;

flex: 0 0 auto;

@media only screen and(max-height: 766px) {

height: 60px;

}

nav {

max-width: 300px; width: 100%;

margin: 0 auto;

}

a.nav { font-size: 20px;

line-height: 25px; text-transform: uppercase;

color: #AFD4EC;

text-shadow: 0px 0px 18px #0C89EC, 0 0 0 #2196F3, 0px 2px 25px #0F71D2; letter-spacing: 2px;

}

}

//end header

.content {

position: relative; flex: 1 0 auto; display: flex; justify-content: center; align-items: center;

overflow: hidden;

}

//pages .pages { position: relative; z-index: 5;

// height: 70vh;

.page {

height: 100%;

position: relative; width: 100%;

&.active { z-index: 99;

}

.holder {

height: 100%;

justify-content: center;

}

}

}

@keyframes shake { from, to {

transform: translate3d(0, 0, 0);

}

10%, 30%, 50%, 70%, 90% { transform: translate3d(-10px, 0, 0); }

20%, 40%, 60%, 80% { transform: translate3d(10px, 0, 0);

}

}

.err {

animation-name: shake; animation-duration: 1s;

}

.empty-err, .data-err, .email-err {

position: absolute; opacity: 0;

top: 110px; right: 0;

left: 0; font-size: 18px; font-family: 'Open Sans'; font-weight: 400; margin: auto; width: 100%; color: red;

text-shadow: 0 0 120px #ec0c0c,0 0 20px #ff0300,0 2px 30px #c1031e; text-align: center;

&.active {

opacity: 1;

}

}

input[type="range"] {

position: relative; height: 5px; background: #AFD4EC; box-shadow: 0 0 18px #0C89EC,0 0 0 #2196F3,0 2px 25px #0F71D2; left: 20px; bottom: -10px; border-radius: 8px;

}

.wheel { position: relative; width: 500px; height: 500px; margin: auto; border-radius: 50%;

border: 2px solid #AFD4EC;

box-shadow: inset 0 0 18px #0C89EC,0 0 0 #2196F3,0 2px 25px #0F71D2;

.circles {

position: absolute; background: #c7edfd; width: 140px; height: 140px; border-radius: 50%; display: flex; justify-content: center;

align-items: center;

#Font > .SpaceAgeCyrillic();

font-size: 22px; letter-spacing: 10px;

box-shadow: inset 0 0 55px rgba(3, 52, 70, 0.93); top: 0px; left: 0px; right: 0px; bottom: 0px; margin: auto;

// transform: translateZ(0); transform: translate3d(0,0,0);

animation: myOrbit 60s linear infinite;

&.small { width: 75px; height: 75px; font-size: 14px; letter-spacing: 2.5px;

}

&--php {

animation-delay: -7.5s;

}

&--css {

animation-delay: -15s;

}

&--javascript {

animation-delay: -22.5s;

}

&--ajax {

animation-delay: -30s;

}

&--cplus {

animation-delay: -37.5s;

}

&--html {

animation-delay: -45s;

}

&--java {

animation-delay: -52.5s;

}

&--bash {

animation-delay: -60s;

}

@keyframes myOrbit { from { transform: rotate(0deg) translateX(250px) rotate(0deg); } to { transform: rotate(360deg) translateX(250px) rotate(-360deg); }

}

}

.main-circle { position: absolute; top: 0px; left: 0px; right: 0px; bottom: 0px; margin: auto; display: flex; justify-content: center; align-items: center; text-align: center; width: 140px; height: 140px;

border-radius: 50%;

#Font > .SpaceAgeCyrillic(); font-size: 18px;

line-height: 25px; text-transform: uppercase;

color: #AFD4EC;

text-shadow: 0px 0px 18px #0C89EC, 0 0 0 #2196F3, 0px 2px 25px #0F71D2; letter-spacing: 2px;

cursor: pointer;

&:before, &:after { content: "";

position: absolute; top: 0px; left: 0px; right: 0px;

|  |  |  |
| --- | --- | --- |
|  |  | bottom: 0px; |
|  |  | margin: auto; |
|  |  | border-radius: 50%; |
|  |  | } |
|  |  | &:hover:before { |
|  |  | box-shadow: 0px 0px 18px #0C89EC, 0 0 0 #2196F3, 0px 2px 25px #0F71D2; |
|  |  | } |
|  |  | &:before { |
|  |  | width: 100%; |
|  |  | height: 100%; |
|  |  | border: 1px dashed #AFD4EC; |
|  |  | } |
|  |  | &:after { |
|  |  | width: 95%; |
|  |  | height: 95%; |
|  |  | border: 1px solid #AFD4EC; |
| } |  | } |
|  |  |  |
|  | @media only screen and(max-height: 766px) { | |
|  |  | width: 400px; |
|  |  | height: 400px; |
|  |  | .circles { |
|  |  | width: 120px; |
|  |  | height: 120px; |
|  |  | &.small { |
|  |  | width: 65px; |
|  |  | height: 65px; |
|  |  | } |
|  |  | @keyframes myOrbit { |
|  |  | from { transform: rotate(0deg) translateX(200px) rotate(0deg); } |
|  |  | to { transform: rotate(360deg) translateX(200px) rotate(-360deg); } |
|  |  | } |
|  |  | } |
|  |  | .main-circle { |
|  |  | width: 105px; |
|  |  | height: 105px; |
|  |  | font-size: 12px; |
|  |  | } |
| } | } |  |

.height {

height: 100%;

// overflow-y: scroll; align-items: center; width: 100%; display: flex; justify-content: center;

}

.user-menu-exit {

position: fixed; right: 15px; top: 10px; width: 80px; height: 70px; z-index: 99;

display: none;

&.show { display: block;

}

a {

display: block;

background: url('../images/exit.png')no-repeat center;

background-size: 30px 30px; width: 80px; height: 70px;

cursor: pointer;

}

}

.user-menu-admin { position: fixed; right: 110px; top: 10px; height: 70px;

z-index: 99;

a {

font-size: 20px; line-height: 70px; cursor: pointer;

color: #fff;

font-family: 'Helvetica';

}

}

.holder { width: 768px; margin-top: 50px;

>div {

margin-bottom: 35px;

}

.title {

display: inline-block; color: #ffffff; font-family: 'Helvetica'; font-size: 18px;

margin-bottom: 15px;

}

.center {

justify-content: center;

}

.admin-form { height: 150px;

margin-bottom: 20px;

&.top {

margin-top: 30px;

.btn-data--add { margin-bottom: 42px;

}

}

.question-wrap {

width: 40%;

textarea { resize: none;

font-size: 18px; width: 100%; height: 115px; border-radius: 3px; border: none; padding: 10px;

background: #c2f6fc;

color: #0d5796;

}

}

.answers-options { justify-content: flex-start; height: 100%;

}

input {

margin-bottom: 13px; display: inline-block; width: 90%; border: none; border-radius: 3px; height: 30px;

line-height: 30px; background: #c2f6fc; color: #0d5796; padding: 0 10px; font-size: 15px;

}

}

.right-ans {

input { margin-bottom: 85px;

}

}

.left { justify-content: flex-start;

span {

margin-right: 20px;

}

input { width: 80px;

}

}

}

.btn-data {

height: 40px; border-radius: 3px; font-size: 17px; font-family: 'Helvetica'; color: #ffffff; padding: 0 10px; line-height: 40px; margin: 0 0 10px; cursor: pointer; width: 90px;

text-align: center;

&--update {

background: #13bc68;

}

&--delete {

background: #ff4646;

}

&--add { background: #fff;

color: #0a8ab4;

}

}

#score {

position: relative; // top: 20px; left: -10px;

font-size: 22px; line-height: 30px; color: #F7D546; // #Font > .SpaceNeon();

text-shadow: 0 0 18px #DFEC0C,0 0 0 #EEF321,0 2px 25px #CDD20F; letter-spacing: 7px;

text-transform: uppercase;

&:before { content: ''; position: absolute; left: -30px;

top: 2px;

background: url('../images/award.png') no-repeat center;

background-size: 20px 20px; width: 30px;

height: 30px;

}

}

footer {

flex: 0 0 auto;

}

#chart{ position:relative; width: 100%;

z-index: 999;

svg {

width: 500px;

}

}

#question{

position: absolute;

visibility: hidden; opacity: 0; width: 100vw; height: 100vh; top: 0; left: 0;

z-index: -1;

// background: rgba(0, 23, 68, 0.87);

background: linear-gradient( 36deg, rgb(14, 140, 36), #175a9e, #175a9e, rgb(14, 140, 36)); transition: all .5s ease;

&.active { display: flex; justify-content: center; visibility: visible; position: fixed; z-index: 9999;

opacity: 1;

}

h1 { max-width: 400px; font-size: 24px; line-height: 30px;

font-weight: bold;

font-family: "Helvetica Neue", Helvetica, Arial, sans-serif; color: #ffffff; text-shadow: none; margin-bottom: 80px; letter-spacing: 2px;

text-align: center;

}

label { position: relative;

font-family: "Helvetica Neue", Helvetica, Arial, sans-serif; font-size: 18px; line-height: 24px; margin: 0 20px; color: #F7D546; min-width: 10px;

&:before, &:after { content: ''; position: absolute; top: -50px;

left: 0;

right: 0;

margin: auto;

}

&:before { width: 30px; height: 30px;

border: 2px solid #ffffff;

border-radius: 4px;

}

span { display: none; position: absolute; top: -45px; left: 0; right: 0; margin: auto; width: 8px;

height: 14px;

border-right: 3px solid #F7D546; border-bottom: 3px solid #F7D546;

transform: rotate(45deg);

}

input[type="radio"]:checked + span {

display: flex;

}

input[type="radio"] { display: none;

}

}

input[type="submit"] { display: block; margin: 30px auto; width: 80px; height: 40px; line-height: 35px; border: 2px solid #ffffff; border-radius: 4px; color: #F7D546; background: none; font-size: 20px; font-weight: 400;

cursor: pointer;

}

form {

text-align: center; width: 600px;

margin: 0 auto;

}

}

.message { position: absolute; display: none; top: 0; left: 0; z-index: 9999; width: 100%; opacity: 0; visibility: hidden; display: flex; text-align: center; align-items: center;

min-height: 100%;

transition: opacity .7s ease;

&.active { display: flex; justify-content: center; position: fixed; opacity: 1;

visibility: visible;

&.success {

// background: rgba(17, 142, 139, 0.99);

background: rgb(36, 154, 169);

}

&.error {

// background: rgba(60, 0, 0, 0.99);

background: rgba(144, 0, 0, 0.99);

}

&.empty-option { background: rgba(144, 0, 0, 0.99);

align-items: center;

}

}

img {

width: 300px;

} p { font-size: 30px; margin-bottom: 40px;

color: #F7D546;

text-shadow: 0 0 28px #DFEC0C,0 0 0 #EEF321,0 2px 42px #CDD20F;

}

.txt { text-shadow: none; text-transform: uppercase;

font-size: 50px;

}

.btn-ok { width: 160px; height: 50px; line-height: 50px; border: 1px solid #ffffff;

color: #F7D546;

text-align: center; font-size: 15px; border-radius: 7px; font-family: 'OpenSans';

text-shadow: 0 0 28px #DFEC0C,0 0 0 #EEF321,0 2px 42px #CDD20F;

box-shadow: inset 0 0 10px #ffffff,0 0px 1px #ffffff; cursor: pointer;

}

}

//carts page

button{

background: #82d2e5; color: #ffffff; border: solid 1px #ffffff; text-decoration:none; cursor: pointer; font-size: 1.2em; padding: 18px 10px; width: 180px; margin: 10px;

outline: none;

&:hover { background:#ffffff; border: solid 1px #ffffff; color: #82d2e5;

transition: all 0.5s ease-in-out;

}

}

.level {

justify-content: center;

align-items: center;

a {

position: relative;

border: 5px solid #AFD4EC;

text-shadow: 0 0 18px #0c89ec,0 0 0 #2196f3,0 2px 25px #0f71d2;

border-radius: 3px; font-size: 30px; line-height: 62px; font-weight: 700; text-align: center; height: 77px; color: #AFD4EC; padding: 0 20px; width: auto;

#Font > .SpaceAgeCyrillic;

}

}

.container {

width: 100%; margin: 20px auto 40px; display: flex;

flex-flow: row wrap;

justify-content: center;

&:after { content: ""; display: table;

clear: both;

}

}

.page--cards {

.time { display: flex;

flex-flow: row; justify-contetn: center; align-items: center; width: 300px; margin: auto;

font-size: 4em;

text-align: center; color: #AFD4EC; text-shadow: 0px 0px 18px #0C89EC, 0 0 0 #2196F3, 0px 2px 25px #0F71D2;

#Font > .SpaceAgeCyrillic(); letter-spacing: 7px;

span{

#Font > .SpaceAgeCyrillic();

color: #AFD4EC;

text-shadow: 0px 0px 18px #0C89EC, 0 0 0 #2196F3, 0px 2px 25px #0F71D2; letter-spacing: 7px;

}

}

label { font-size: 20px;

color: #AFD4EC;

font-family: 'Helvetica', sans-serif;

color: #AFD4EC;

text-shadow: 0px 0px 18px #0C89EC, 0 0 0 #2196F3, 0px 2px 25px #0F71D2;

}

}

#container-q {

[data-view="card"] xmp{

font-size: 25px;

}

}

[data-view="card"] { display: flex; justify-content: center;

align-items: center; width: 180px; height: 180px; border: solid 1px #d3cece; border-radius: 5px; margin: 5px;

background: linear-gradient(135deg, #f3f3f3 22px, #ffffff 22px, #ffffff 24px, transparent 24px, transparent 67px, #ffffff 67px, #ffffff 69px, transparent 69px),linear-gradient(225deg, #f3f3f3 22px, #ffffff 22px, #ffffff 24px, transparent 24px, transparent 67px, #ffffff 67px, #ffffff 69px, transparent

69px)0 64px;

background-color: #f3f3f3; background-size: 64px 128px; transform: rotateY(180deg); cursor: pointer;

xmp {

text-align: center;

font-weight: 700;

color: #1f7690; opacity: 0; white-space: pre-wrap; transition: all .6s ease;

}

&.flipped {

// transform-style: preserve-3d; position: relative; transform: rotateY(0deg);

transition: 0.6s;

xmp {

opacity: 1;

}

}

}

.reverse {

transition: 0.6s; transform-style: preserve-3d; position: relative;

transform: rotateY(180deg);

}

.correct {

opacity: .5; cursor: default; transform-style: preserve-3d; position: relative;

transform: rotateY(0deg);

}

.statistics {

width: 580px;

margin: auto;

p , span{ color: #afd4ec; #Font >.SpaceAgeCyrillic(); font-size: 49px; line-height: 51px; letter-spacing: 5px;

text-align: center;

text-shadow: 0 0 18px #0c89ec, 0 0 0 #2196f3, 0 2px 25px #0f71d2; margin-bottom: 40px;

}

span { color: #F7D546;

text-shadow: 0 0 18px #C1C939,0 0 0 #EEF321,0 2px 25px #CDD20F;

}

.score {

position: relative; margin: auto; display: flex;

justify-content: center; font-size: 40px; text-align: center; line-height: 50px; margin-bottom: 40px;

color: #F7D546;

text-shadow: 0 0 18px #C1C939,0 0 0 #EEF321,0 2px 25px #CDD20F; letter-spacing: 5px;

text-transform: uppercase;

}

}

flex.css

.flex {

display: flex;

align-items: center;

&--center {

justify-content: center;

}

&--start {

justify-content: flex-start;

}

&--end {

justify-content: flex-end;

}

&--between { justify-content: space-between;

}

&--around {

justify-content: space-around;

}

}

.row {

flex-flow: row;

}

.column { flex-flow: column;

}

**reset.css**

/\* Eric Meyer's Reset CSS v2.0 - http: //cssreset.com \*/ html,body,div,span,applet,object,iframe,h1,h2,h3,h4,h5,h6,p,blockquote,pre,a,abbr,acronym,address,big, cite,code,del,dfn,em,img,ins,kbd,q,s,samp,small,strike,strong,sub,sup,tt,var,b,u,i,center,dl,dt,dd,ol,ul,li,fi eldset,form,label,legend,table,caption,tbody,tfoot,thead,tr,th,td,article,aside,canvas,details,embed,figure, figcaption,footer,header,hgroup,menu,nav,output,ruby,section,summary,time,mark,audio,video{border:

0;font-size: 100%;font: inherit;vertical-align: baseline;margin: 0;padding:

0}article,aside,details,figcaption,figure,footer,header,hgroup,menu,nav,section{display: block}body{line-height: 1}ol,ul{list-style: none}blockquote,q{quotes: none}blockquote:before,blockquote:after,q:before,q:after{content: none}table{border-collapse: collapse;border-spacing: 0}

html {

box-sizing: border-box;

}

\*, \*:before, \*:after {

box-sizing: inherit;

}

\* {

margin: 0; padding: 0;

outline: none;

}

html, body { position: relative; width: 100%;

min-height: 100%;

}

button, input[type=submit], input[type=button], .submit, .btn {

-webkit-touch-callout: none; user-select: none;

}

input, textarea {

font-family: inherit;

}

::placeholder {

color: inherit;

opacity: 1;

}

em { font-style: italic;

}

a {

outline: none; text-decoration: none;

-webkit-tap-highlight-color: rgba(0,0,0,0);

}

p:last-child {

padding-bottom: 0;

margin-bottom: 0;

}

hr {

display: block; height: 2px; border: none; padding: 0; clear: both;

}

svg { display: inline-block; overflow: hidden;

width: 100%;

}

fonts.css

#Font {

.Space() { font-family: 'Space';

}

.SpaceNeon() { font-family: 'SpaceNeon';

}

.SpaceAgeCyrillic() {

font-family: 'SpaceAgeCyrillic';

}

}

@font-face { font-family: 'Space'; src: url('../../../assets/front/fonts/spac3\_tech\_free\_promo-webfont.woff2') format('woff2'),

url('../../../assets/front/fonts/spac3\_tech\_free\_promo-webfont.woff') format('woff'), url('../../../assets/front/fonts/spac3\_tech\_free\_promo-webfont.ttf') format('truetype'); font-weight: normal;

font-style: normal;

}

@font-face {

font-family: 'SpaceNeon';

src: url('../../../assets/front/fonts/spac3\_neon\_free\_promo-webfont.woff2') format('woff2'), url('../../../assets/front/fonts/spac3\_neon\_free\_promo-webfont.woff') format('woff'), url('../../../assets/front/fonts/Spac3\_neon free promo.ttf') format('truetype'); font-weight: normal;

font-style: normal;

}

@font-face {

font-family: 'SpaceAgeCyrillic';

src: url('../../../assets/front/fonts/space\_age\_cyrillic-webfont.woff2') format('woff2'), url('../../../assets/front/fonts/space\_age\_cyrillic-webfont.woff') format('woff'), url('../../../assets/front/fonts/space\_age\_cyrillic.ttf') format('truetype'); font-weight: normal;

font-style: normal;

}

connection.php <?php

$db = new PDO("mysql:host=localhost;dbname=lecode;port=33060", "homestead", "secret"); ?>

Използвани библиотеки:

angular.audio.js angular.js angular-route.js angular-resource.js

d3.js stopExecutionOnTimeout.js