



Софийски университет „Св. Кл. Охридски”

Факултет по математика и информатика

Катедра „Софтуерни технологии”



ДИПЛОМНА РАБОТА

на тема

„Софтуерна платформа за анализ и
визуализация на резултати от видео
игри за обучение”

Дипломант: **Мартин Андонов Ковачев**

Специалност: **Софтуерни технологии**

Факултетен номер: **M26219**

Научен ръководител:

проф. д-р Боян Бончев

София, 2021 г.

Съдържание

Глава 1. Увод	3
1.1. Актуалност на проблема и мотивация	3
1.2. Цел и задачи на дипломната работа	4
1.3. Очаквани ползи от реализацията	4
1.4. Структура на дипломната работа	4
 Глава 2. Преглед на софтуерни платформи за анализ и визуализация на резултати от видео игри за обучение	 6
2.1. Основни дефиниции	6
2.2. Основни проблеми при анализ и визуализация на резултати от видео игри за обучение	9
2.3. Подходи за решаването на проблемите	10
2.4. Съществуващи решения	10
2.5. Избор на критерии за сравнение и сравнителен анализ на решенията	16
2.6. Изводи	19
 Глава 3. Използвани технологии и платформи	 21
3.1. Изисквания към средствата	21
3.2. Видове средства и начин и място за използването им – сравнителен анализ	22
3.3. Избор на средствата	41
3.4. Изводи	42
 Глава 4. Анализ	 43
4.1. Концептуален модел	43
4.2. Потребителски изисквания	45
4.3. Качествени изисквания	46
4.4. Работни процеси	49
4.5. Изводи	54
 Глава 5. Проектиране	 56
5.1. Обща архитектура	56
5.2. Модел на данните	61
5.3. Диаграми (на структура и поведение - по слоеве и модули, с извадки от кода)	64
5.4. Потребителски интерфейс	67
 Глава 6. Реализация, тестване/експерименти и внедряване	 72
6.1. Реализация на модулите	72
6.2. Планиране на тестването	81
6.3. Модулно и системно тестване	81
6.4. Анализ на резултатите от тестването и начин на отразяването им	83
6.5. Експериментално внедряване	84
6.6. Анкета	84

Глава 7. Заключение	87
7.1. Обобщение на изпълнението на началните цели	87
7.2. Насоки за бъдещо развитие и усъвършенстване	88
 Използвана литература	 89

Глава 1. Увод

1.1. Актуалност на проблема и мотивация

Компютърните игри навлизат все повече в живота на хора от различни възрасти. Те се разпространяват в много нови области, различни от развлечението. Един от най-успешните примери е в сферата на образованието и обучението. Този тип игри се нарича сериозни игри и той набира все по-голяма популярност сред съвременното общество. Определено се счита, че с развитието на технологиите потребителските изисквания на потребителите на образователни видео игри непрекъснато се променят. Това е една от причините, които определят необходимостта от интегриране и използване на инструменти за наблюдение на информация, данни, дейности, свързани с процесите на създаване, проектиране, управление и играене на образователни видео игри.

Дипломната работа е в областта на софтуерните платформи за анализ и визуализация на резултати от играни видео игри за обучение, предназначени за използване както от заинтересовани лица като проектантите на видео игри (напр. дизайнери и програмисти), педагози и учители, така и от самите играчи (ученици). Тя цели да създаде експериментална софтуерна онлайн платформа за анализ и визуализация на резултати от играни видео игри за обучение, което включва представяне и сортиране по критерии на учебни и игрови резултати от играни конкретни игри от даден играч, както и статистически анализ на тези резултати, евентуално съотнесени към характеристиките на модела на играча, например изчисляване на корелации между резултатите за различни учебни видео игри и модела на играча. Тази платформа ще се интегрира към софтуерната система, разработвана в рамките на проект АПОГЕЙ (<http://apogee.online/>). Използването на такова приложение ще даде възможност на различните лица, заинтересовани в процеса на създаване на видео игри, да се информират за резултатите от играните учебни игри в платформата на проект АПОГЕЙ, както и техните взаимовръзки. [\[1\]](#)

1.2. Цел и задачи на дипломната работа

Целта на дипломната работа е да се проектира, разработи и тества експериментална софтуерна онлайн платформа за анализ и визуализация на резултати от играни видео игри за обучение, разработвани в рамките на проекта АПОГЕЙ (<http://apogee.online/>). Приложението трябва да предоставя възможност за анализ и визуализация на големи масиви от потребителски данни.

Задачи, произтичащи от целта, са следните:

1. Преглед на съществуващите платформи за анализ и визуализация на потребителски данни.
2. Преглед на използвани софтуерни технологии и платформи за разработка на приложения за анализ и визуализация на потребителски данни.
3. Разработване на експериментална софтуерна онлайн платформа за анализ и визуализация на резултати от играни видео игри за обучение, което включва:
 - анализ на изискванията
 - проектиране на решението
 - имплементация на приложението и интеграция с платформата за генериране на игри на проект АПОГЕЙ
 - тестване и внедряване
4. Практически експеримент с разработеното онлайн приложение за анализ и визуализация на резултати от видео игри за обучение, последван от анкетиране на потребителите относно използваемостта на приложението.
5. Анализ на резултатите от експеримента и валидиране на разработеното онлайн приложение за анализ и визуализация на резултати от видео игри за обучение.

1.3. Очаквани ползи от реализацията

Ползата от такова решение би била много голяма, понеже то би представлявало безплатна платформа за анализ и визуализация на резултати от играни видео игри за обучение, разработвани в рамките на проекта АПОГЕЙ. То би заместило съществуващия понастоящем процес на представяне и анализ на резултатите в таблици на Microsoft Excel.

1.4. Структура на дипломната работа

В първа глава на дипломната работа се разглежда цялостната същност на проблема. Представена е актуалността на проблема, какви цели и задачи трябва да бъдат постигнати, за да може да се реализира и какви са очакваните ползи от реализацията на подобна платформа.

Във втора глава на дипломната работа са засегнати основните проблеми при анализа и визуализацията на резултати от играни видео игри за обучение и подходите за решението им. Представени са съществуващи софтуерни продукти (среди) за анализ и обработка на данни. Също така е направен сравнителен анализ между съществуващите решения.

В трета глава се разглеждат и сравняват различни технологии за реализиране на софтуерния продукт. Направен е сравнителен анализ между тях и са избрани най-подходящите за реализиране на системата.

В четвърта глава се разглежда концептуалния модел на системата. Описани са потребителските и качествените изисквания, правата, ролите, също така и диаграми описващи бизнес процесите в системата.

В пета глава е описано проектирането на системата. Разгледана е архитектурата и се анализират нейните слоеве. Уточнява се модела на данните и по какъв начин ще бъдат представени. Представят се структурни диаграми и диаграми на поведението. Разглежда се потребителският интерфейс.

В шеста глава се представя реализирането на модулите и се анализират сценариите за тестване, и се разглеждат резултатите от тестването на системата. Описан е и начинът за внедряване на системата. Накрая са включени и резултатите от анкетата за оценка на самата система.

В последната глава на дипломната работа е направено заключение на постигнатите резултати и какви биха били бъдещите стъпки за развитие на платформата за визуализация и анализ на резултати от играни видео игри за обучение.

Глава 2. Преглед на софтуерни платформи за анализ и визуализация на резултати от видео игри за обучение

2.1. Основни дефиниции

Компютърните игри навлизат все повече в живота на хора от различни възрасти. Те се разпространяват в много нови области, различни от развлечението. Един от най-успешните примери е в сферата на образованието и обучението. Този тип игри се нарича сериозни игри и той набира все по-голяма популярност сред съвременното общество. Сериозните игри за обучение представляват съвременен начин за усвояване на нови умения, като редица сектори значително са увеличили тяхната употреба, част от тях са образование, отбрана, здраве и наука. С развитието на технологиите изискванията на потребителите на образователни видео игри непрекъснато се променят, определяйки необходимостта от интеграция и използване на инструменти за следене на информация, данни, дейности, свързани с процеси на създаване, проектиране, управление и възпроизвеждане на образователни видео игри. [\[2\]](#)

Терминът сериозни игри за първи път е използван от Кларк Абт през 1970 г. [\[2\]](#) Това понятие включва настолни и дигитални игри. Според Абт игрите имат изрично заявена и внимателно обмислена образователна цел, и не са предназначени да се играят предимно за забавление. Това не означава, че сериозните игри не са или не трябва да са забавни. Разработената от него игра T.E.M.P.E.R. представя възможните последици от Студената война в световен мащаб. Абт счита, че игрите мотивират обучаемите, които имат умения, но не са обучавани правилно. Първата обучаваща компютърна игра „The Oregon Trail“, разработка на Minnesota Educational Computing Consortium (MECC), през 1971 г. е изцяло текстова. Създадените през следващите години компютърни игри са главно с цел забавление и за комерсиални цели. С развитието на компютърните технологии наред с комерсиалните игри се появяват и игри с образователна цел в сферата на културата и изкуството като „Versailles 1685“ – 1997 г., „China: The Forbidden City 1775“ – 1998 г., „Egypt 1156BC Tomb of the Pharaoh“ – 1999 г. и други [\[2\]](#). Развитието на сериозните игри започва с първата станала популярна съвременна видео игра „America’s Army“ през 2002 г. [\[2\]](#) Вероятно голямата популярност на „America’s Army“ е причина най-широко приетата дефиниция на термина да принадлежи на един от нейните създатели Michael Zyda. Той определя сериозните игри като забавление със задължителна възпитателна съставка: „Сериозна игра е интелектуално състезание, играе се на компютър, в

съответствие със специфични правила. Елементът на забавление се използва с цел управление на правителствено или фирмено обучение, образование, здравеопазване, обществен ред, стратегическите цели за комуникация“. [\[3\]](#)

При видео игрите за обучение е изключително важно да се събират данни за поведението и начина на игра на играчите. Статистиката ни помага да използваме правилните методи за събиране на данни, да използваме правилните анализи и да представим ефективно резултатите. Статистиката е решаващ процес зад начина, по който правим открития в науката, вземаме решения въз основа на данни и правим прогнози. Статистиката ни позволява да разберем дадена тема много по-дълбоко.

Използваните в дипломната работа дескриптивни статистики, метрики и методи за статистически анализ са както следва:

- **Стандартно отклонение**

- *Стандартното отклонение* е квадратен корен на дисперсията, изчислено в същите мерни единици като променливата. *Дисперсията* е сумата на квадратите на девиациите в разпределението относно неговата средна стойност. *Девиация* е разликата между дадено наблюдение от извадката и нейната средна стойност. [\[4\]](#)

- Дадена е извадка от числа: 20, 21, 23, 25 и 26
- Средната стойност е 23
- Девиацията за първото наблюдение е $|20 - 23| = 3$ или за редицата е 3, 2, 0, 2, 3
- Дисперсията за първото наблюдение е $3^2 = 9$ или $9 + 4 + 0 + 4 + 9) / 5 = 5.2$
- Стандартно отклонение $\sqrt{5.2} = 2.28$

- **Стандартна грешка**

- *Стандартната грешка* е мярка за променливостта на извадката, изчислена въз основа на стандартното отклонение на популацията.
 - Стандартна грешка = Стандартно отклонение / \sqrt{n} , където n е брой на елементите в множеството

- **Корелация на Пирсън**

- *Корелацията* е мярка за силата на зависимостта между две променливи. *Корелационен анализ* е статистически анализ за проверка на хипотезата за (не случайна) връзка между променливите. *Корелация на Пирсън* е метрика, оценяваща силата на линейната връзка между две променливи. Обозначава се със

символа r и винаги е в интервала $[-1.00, +1.00]$. Колкото корелационният коефициент на Пирсън е по-близо до абсолютна стойност до 1.00, толкова линейната връзка между двете променливи е по-силна. Колкото обаче корелационният коефициент на Пирсън е по-близо до 0, толкова е по-слаба линейната връзка между двете променливи.

- **Т-тест на Стюдънт**

- *Т-тест на Стюдънт* е параметричен статистически тест, който се използва при проверка дали разликата между две средноаритметични стойности на две различни извадки от данни е статистически значима или се дължи на случайни фактори, или на по-малък брой данни.
- *Нулевата хипотеза* H_0 , която трябва да се провери, е твърдение, че няма различие или взаимовръзка. Нулевата хипотеза H_0 е взаимно изключваща се с алтернативната (експерименталната) хипотеза H_1 . Ако е изпълнена H_0 означава, че прогнозата не е вярна и че прогнозираният ефект не съществува.
- Доказателствата, които се използват, за да се отхвърли нулевата хипотеза H_0 за липса на различия се сумират във вероятност, която се нарича *p-стойност*. Колкото е по-малка *p-стойността*, толкова са повече доказателствата, с които изследователят разполага, за да отхвърли нулевата хипотеза H_0 . [5]

- **Дисперсионен анализ ANOVA (Analysis of variance)**

- В дисперсионния анализ чрез сравняване на средноаритметичните стойности за извадките се правят изводи за съотношенията между средните стойности на генералната съвкупност. [6]

- **Размер на ефекта на Коен (Effect size)**

- Коен d или стандартизирана средна разлика е един от най-често срещаните начини за измерване на размера на ефекта. Размерът на ефекта е колко голям е ефектът. Например, лекарството А има по-голям ефект от лекарството В. Въпреки че *p-стойността* може да ви каже дали има ефект, няма да ви каже колко голям е този ефект. [7]

Анализът и по-конкретно обучителният анализ включва измерване, компилиране, и анализ на данни с цел оптимизиране на резултатите от обучението. Анализът по същество е „практическа информация“, което означава, че ви позволява да разгледате по-задълбочено цялостното представяне на една сериозна игра и да определите дали тя отговаря на вашата учебна цел или не.

Събраната информация показва кои задачи и предизвикателства помагат на учащите да постигнат общата цел, и кои области могат да се подобрят. Има различни методи за събиране на данни, като всички те ще ви дадат по-ясна перспектива за това как се представя играта ви и как вашата аудитория взаимодейства с нея. Колкото повече данни имате на ваше разположение, толкова по-голяма е вероятността да вземате решения, които ще гарантират успех.

2.2 Основни проблеми при анализ и визуализация на резултати от видео игри за обучение

Един от основните проблеми при анализа и визуализацията на резултати от видео игри за обучение е липсата на добре разработени и лесни за работа програмни среди и средства, които да предоставят онлайн възможност и реактивност за анализ и визуализация на резултатите. Най - често срещаните среди за обработка на този тип данни са системи като Excel, MatLab, R, SPSS и други. Основният проблем при този тип системи е, че те не са специализирани за анализ и обработка на резултати от играни видео игри, а по-скоро решение, чрез което може да се достигне исканият краен резултат, без да се взима предвид времето отделено за вписване на данните и цялостната интеграция. По този начин има огромна загуба на време и голям шанс за допускане на грешки, които могат да доведат до неправилни крайни резултати от изследванията. Трудно е да бъде направено общо решение на този вид проблем, като се има предвид различните видове изследвания, които трябва да бъдат направени за различните видове игри.

С цел ефикасна и безпроблемна обработка на резултатите от видео игрите за обучение, такива програмни среди и средства би трябвало да бъдат интегрирани в самата система за управление на видео игрите за обучение. При тази интеграция програмното средство трябва да познава произхода, типа и семантиката на данните, получени като игрови метрики (например: време, точки, ефективност, ефикасност и други) от система за управление при игровите сесии на различните в системата регистрирани играчи. Докато произходът на данните касае начина на изпълнението на учебната задача за дадена игра за обучение, то семантиката им има отношение към начина на играене или *game playability* и възможността за учене чрез игри или *learnability*. По този начин ще могат да се правят изключително точни резултати и да се проследяват играчите до какво ново усвояват учебните задачи.

Програмните среди за анализ и визуализация на резултати от видео игри за обучение би трябвало да поддържат освен дескриптивни статистики (средно,

стандартно отклонение и стандартна грешка) и разнообразни метрики и методи за статистически анализ на резултатите от игровите сесии, като корелация на Пирсън, размер на ефекта на Коен, Т-тест на Стюдънт и други.

Програмните средства за анализ и визуализация на резултати от видео игри за обучение би трябвало да позволяват лесно разширение с нови метрики и методи за статистически анализ на резултатите от игровите сесии, което да става лесно, без да е нужно рестартиране на системата. Също така, да позволява лесно редактиране на данните от базата данни при възникнали проблеми.

От огромно значение е и графичният интерфейс на програмните средства да е интуитивен, лесен и прост за работа, позволяващ и на не-ИКТ специалисти да работят с тях.

2.3 Подходи за решаването на проблемите

Реализиране на уеб-базирано приложение за анализ и визуализация на резултати от играни видео игри за обучение е решение на дадените проблеми. Регистрираните играчи могат лесно и бързо да влязат в системата чрез техните имейли и пароли и да проследят резултатите си чрез навигиране в правилните менюта на приложението. По този начин от компютър или телефон, играчите ще могат да виждат генерираните от системата резултати от различните статистически анализи. Системата сама ще събира данните, без да е нужна никаква намеса от играчите. Хронология на изиграните игри и прилежащите към тях игрови метрики (време, ефективност, ефикасност и други) ще бъдат налични за всеки един играч. От страна на изследователите, приложението ще замени изцяло нуждата от използване на системи като Excel, MatLab, R, SPSS и други и нуждата да се въвеждат данните ръчно. Резултатите ще се извеждат и в графичен вид, за да се проследят изменения в поведението на някои метрики.

2.4 Съществуващи решения

Видео игрите за обучение набират все по-голяма популярност сред хората и все повече информация бива събирана и анализирана. Изследването на поведението на играчите става все по-задълбочено с развитието на новите технологии. Изкуственият интелект допринася за намиране на поведения в големи масиви от данни, които хората не могат да забележат. Машинното обучение позволява на машините да обработват и анализират информация, а в някои случаи да се учат сами по много сложен начин.

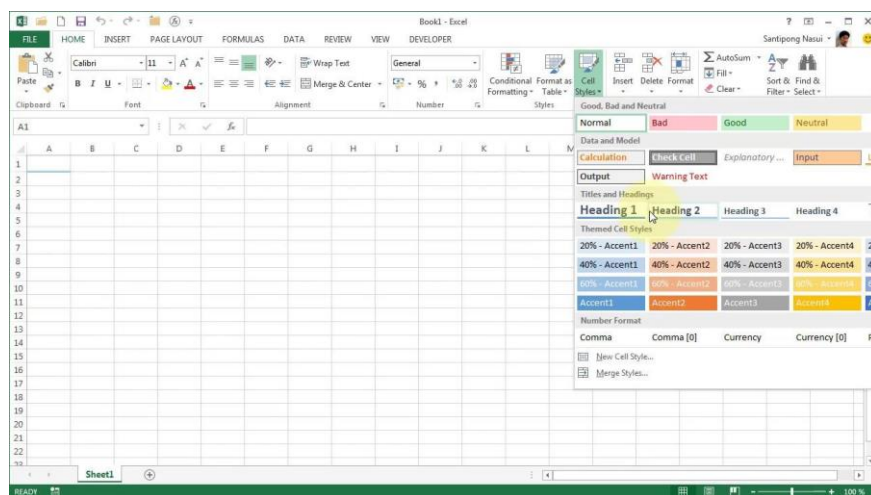
С развитието на уеб технологиите, много от компаниите започват да предлагат софтуерните си продукти и под формата на уеб приложения. По този начин се избягва нуждата от инсталиране на софтуер на реална машина и може да се използва от всяко устройство с връзка към интернет без значение от местоположението на потребителя. Тук ще разгледаме част от най-известните софтуерни продукти (среди) за анализ и обработка на данни и ще представим кратко ревю на всеки един от тях.

Excel

Microsoft Excel е полезна и мощна програма за анализ на данни и документиране Фиг. 1. Това е програма за електронни таблици, която съдържа редица колони и редове, където всяко пресичане на колона и ред е „клетка“. Всяка клетка съдържа една точка от данни или една информация. Като организирате информацията по този начин, можете да улесните намирането на информация и автоматично да извлечете информация от променящите се данни. [8]

Основните приложения на Excel включват:

- Въвеждане на данни
- Управление на данни
- Счетоводство
- Финансов анализ
- Диаграми и графики
- Програмиране
- Управление на времето
- Управление на задачи
- Финансово моделиране
- Управление на взаимоотношенията с клиенти (CRM)



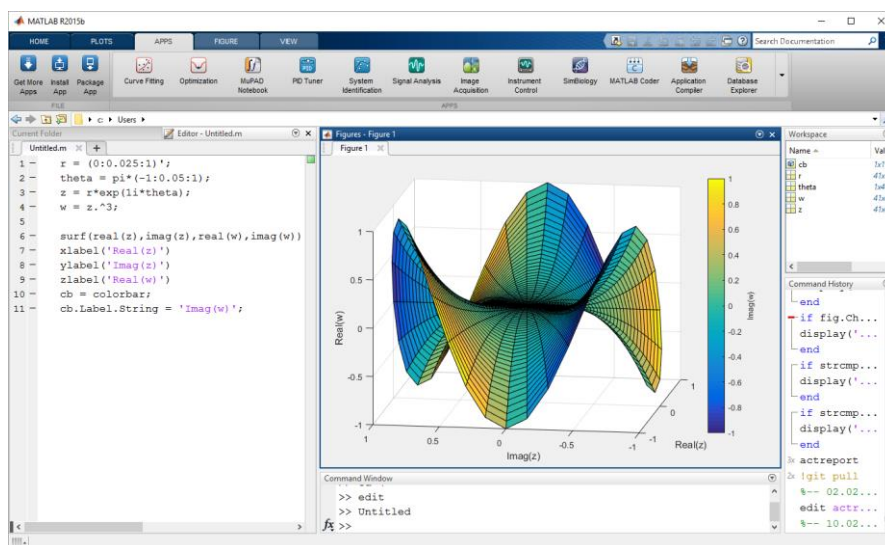
Фиг. 1 Изглед на софтуерния продукт Excel

MatLab

MATLAB® е платформа за програмиране, предназначена специално за инженери и учени да анализират и проектират системи и продукти, които трансформират нашия свят Фиг. 2. Сърцето на MATLAB е езикът MATLAB, базиран на матричен език, позволяващ най-естественото изразяване на изчислителната математика. Милиони инженери и учени по целия свят използват MATLAB за редица приложения в индустрията и академичните среди, включително дълбоко обучение и машинно обучение, обработка на сигнали и комуникации, обработка на изображения и видео, системи за управление, тестове и измервания, изчислително финансиране и изчислителна биология. [9]

MATLAB предоставя:

- Типове данни и възможности за предварителна обработка, предназначени за инженерни и научни данни
- Интерактивни и персонализирани визуализации на данни
- Хиляди предварително изградени функции за статистически анализ, машинно обучение и обработка на сигнали
- Обширна и професионално написана документация
- Ускорена производителност с прости промени в кода и допълнителен хардуер
- Разширен анализ до големи данни без големи промени в кода
- Автоматично пакетиране на анализ в свободно разпространяващи се софтуерни компоненти или вграден изходен код без ръчно прекодиране на алгоритми
- Автоматично генерирани от вашия анализ отчети за споделяне



Фиг. 2 Изглед на софтуерния продукт MatLab

R

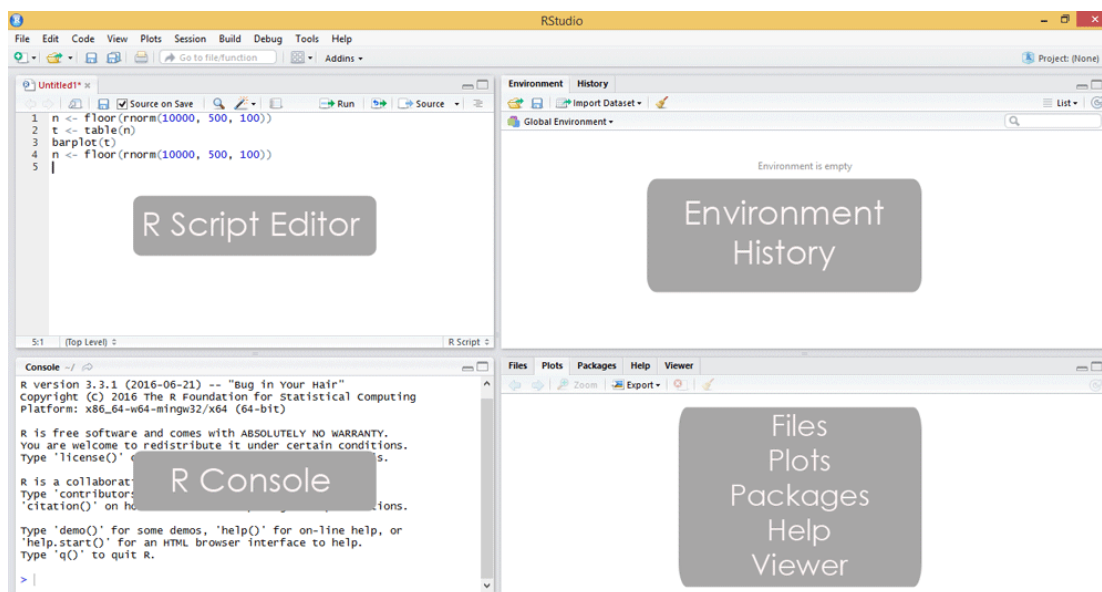
R е език и среда за статистически изчисления и графики. Това е проект на GNU, който е подобен на езика и околната среда S, разработен в Bell Laboratories (преди AT&T, сега Lucent Technologies) от Джон Чембърс и колегите му Фиг. 3. R може да се разглежда като различна реализация на S. Има някои важни разлики, но много код, написан за S, остава непроменен под R. [\[10\]](#)

R предоставя голямо разнообразие от статистически (линейно и нелинейно моделиране, класически статистически тестове, анализ на времеви редове, класификация, групиране,...) и графични техники и е силно разширяем. Езикът S често е средство за избор за изследване на статистическата методология, а R осигурява отворен код за участие в тази дейност. Една от силните страни на R е лекотата, с която могат да се създадат добре проектирани графики с голямо качество при публикуване, включително математически символи и формули, когато е необходимо. Големи грижи са положени по подразбиране за малките дизайнерски избори в графиката, но потребителят запазва пълен контрол.

R е достъпен като свободен софтуер при условията на Общия публичен лиценз на GNU на Фондацията за свободен софтуер под формата на изходен код. Той компилира и работи на голямо разнообразие от UNIX платформи и подобни системи (включително FreeBSD и Linux), Windows и MacOS.

R е интегриран набор от софтуерни съоръжения за обработка на данни, изчисления и визуализиране на графики. Той включва:

- ефективно средство за обработка и съхранение на данни
- набор от оператори за изчисления на масиви, по-специално матрици
- голяма, последователна, интегрирана колекция от междинни инструменти за анализ на данни
- графични средства за анализ на данни и показване на екрана или на хартиен носител
- добре разработен, прост и ефективен език за програмиране, който включва условия, цикли, дефинирани от потребителя рекурсивни функции и средства за въвеждане и извеждане



Фиг. 3 Изглед на среда за разработка (RStudio) чрез езика R

SPSS

SPSS е съкращение от Статистически пакет за социалните науки и се използва от различни видове изследователи за сложен анализ на статистически данни. Софтуерният пакет SPSS е създаден за управление и статистически анализ на данните от социалните науки Фиг. 4. Първоначално е лансиран през 1968 г. от SPSS Inc., а по -късно е придобит от IBM през 2009 г.

Официално наречен IBM SPSS Statistics, повечето потребители все още го наричат SPSS. Като световен стандарт за анализ на данните в областта на социалните науки, SPSS е много желан поради своя ясен и подобен на английски език за командване и впечатляващо подробно ръководство за потребителя. [?]

SPSS се използва от пазарни изследователи, здравни изследователи, компании за проучване, държавни институции, изследователи в образованието, маркетингови организации, анализатори на данни и много други за обработка и анализ на данни от проучванията, като например тези, които събираме с онлайн платформа за проучване като Alchemer. Повечето водещи изследователски агенции използват SPSS, за да анализират данните от анкетите и да извличат текстови данни, така че да могат да извлекат максимума от своите научноизследователски и проучвателни проекти.

SPSS предлага четири програми, които помагат на изследователите със сложни нужди от анализ на данни.

Статистическа програма

Програмата за статистика на SPSS предоставя множество основни статистически функции, някои от които включват честоти, кръстосани таблици и двумерни статистически данни.

Програма за моделиране

Програмата Modeler на SPSS позволява на изследователите да изграждат и валидират модели за прогнозиране, използвайки усъвършенствани статистически процедури.

Програма за текстови анализи за проучвания

Програмата SPSS Text Analytics for Surveys помага на администраторите на анкетите да открият важна информация от отговорите на отворените въпроси на анкетата.

Дизайнер на визуализация

Програмата за визуализация на SPSS Visualization Designer позволява на изследователите да използват своите данни, за да създават голямо разнообразие от визуализации като диаграми на плътността и радиални полета от техните данни от проучване с лекота.

В допълнение към четирите програми, споменати по-горе, SPSS предоставя и решения за управление на данни, които позволяват на изследователите да извършват подбор на случаи, да създават производни данни и да извършват преоформяне на файлове.

SPSS предлага и документация за данни, която позволява на изследователите да съхраняват речник на метаданни. Този речник с метаданни действа като централизирано хранилище на информация, свързана с данните, като значение, взаимоотношения с други данни, произход, употреба и формат.

Има няколко статистически метода, които могат да бъдат използвани в SPSS, включително:

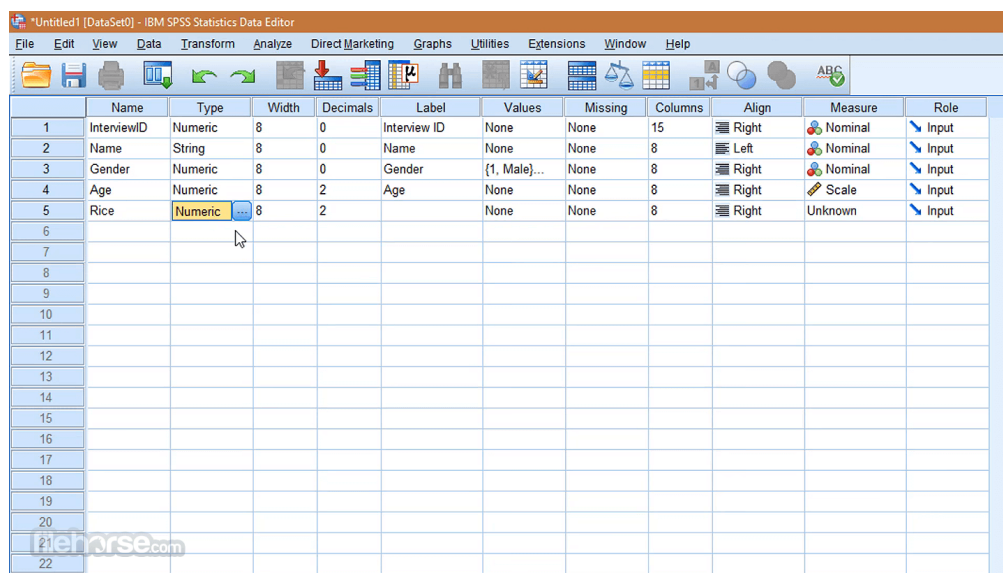
- Дескриптивна статистика, включително методологии като честоти, кръстосани таблици и статистически описателни съотношения.
- Двумерна статистика, включително методологии като анализ на дисперсията (ANOVA), средства, корелация и непараметрични тестове.
- Прогнозиране на числови резултати като линейна регресия.
- Предсказване за идентифициране на групи, включително методологии като клъстер анализ и фактор анализ.

Ползите от използването на SPSS за анализ на данните от проучванията, са следните:

- Благодарение на акцента си върху анализа на статистически данни, SPSS е изключително мощен инструмент за манипулиране и дешифриране на данните от проучванията.
- Експортирането на данни от проучването от Alchemer към собствения .SAV формат на SPSS прави процеса на извличане, манипулиране и анализ на данни чист и лесен. Използвайки формата .SAV, SPSS автоматично настройва и импортира определените имена на променливи, типове променливи, заглавия и етикети на стойности, което прави процеса много по -лесен за изследователите.

След като данните от проучването се експортират в SPSS, възможностите за статистически анализ са практически безкрайни.

Накратко, не забравяйте да използвате SPSS, когато имате нужда от гъвкав, персонализиран начин да получите супер подробно дори и най -сложните набори от данни. Това дава на изследователите, повече време да идентифицират тенденции, да разработват модели за прогнозиране и да правят качествени заключения. [\[11\]](#)



	Name	Type	Width	Decimals	Label	Values	Missing	Columns	Align	Measure	Role
1	InterviewID	Numeric	8	0	Interview ID	None	None	15	Right	Nominal	Input
2	Name	String	8	0	Name	None	None	8	Left	Nominal	Input
3	Gender	Numeric	8	0	Gender	{1, Male}...	None	8	Right	Nominal	Input
4	Age	Numeric	8	2	Age	None	None	8	Right	Scale	Input
5	Rice	Numeric	8	2		None	None	8	Right	Unknown	Input
6											
7											
8											
9											
10											
11											
12											
13											
14											
15											
16											
17											
18											
19											
20											
21											
22											

Фиг. 4 Изглед на софтуерния продукт IBM SPSS

2.5 Избор на критерии за сравнение и сравнителен анализ на решенията

Софтуерните продукти ще бъдат разгледани по няколко критерии:

1. Брой потребители
2. Функционалности
3. Цена
4. Трудност за научаване (крива на учене)

Причината за избор на тези критерии е фокусирането върху анализа на данни, тяхната обработка и визуализация.

Функционалностите, които предлага, и трудността при използването им са несъмнено едни от основните критерии по които даден продукт бива избран при използването им за анализиране на резултати от играни видео игри за обучение. Големият брой функционалности предоставя на крайният потребител по-голямо поле от възможности и да се проследи цялостното представяне на самата сериозна игра.

Останалите критерии са пряко свързани с основните цели на даден софтуерен продукт да бъде разгледан, като предпочитан сред останалите. Популярността е неизменна част от избора на какъвто и да е било продукт и несъмнено трябва да бъде отбелязан в дадения сравнителен анализ на решенията. Тук могат да се представи броят потребители на всеки един продукт.

Някои от продуктите се предоставят в безплатни версии, други имат цена за различни пакети на софтуерния продукт, като това също оказва влияние при избора. В таблицата по-долу ще бъдат представени всички избрани продукти и ще се направи нагледно сравнение между техните възможности.

Табл. 1: Сравнение на съществуващи решения

Софтуерен продукт	Предимства	Недостатъци	Цена	Трудност за научаване	Сфера
Excel	Лесен за научаване, създаване и обработка на таблици, диаграми. описателна статистика, Data Analysis tool pack	Труден за работа с големи количества от данни	Платен, Според избрания план	Ниска	Бизнес

MatLab	Инструменти за обработка на изображения, Матрични манипулации, Статистически анализ	Висока цена	Платен, Според изборния лиценз	Средна	Наука, Инженерство
R	Фокусиран върху статистически анализ и обработка на данни	Труден за научаване	Безплатен, Отворен код	Висока	Финанси, Наука, Статистика
IBM SPSS	Създаване на таблици и диаграми, описателна статистика	Висока цена	Платен, Според изборния план	Средна	Наука, Статистика

1. MatLab и R

MATLAB е език за програмиране, посветен на математическите и техническите изчисления, предназначен за инженери и учени. Настолната среда има естествен начин за изразяване на изчислителна математика като линейна алгебра, анализ на данни, обработка на сигнали и изображения. MATLAB предлага специфично за приложението решение, наречено „Кутии с инструменти“. Кутиите с инструменти предоставят набор от MATLAB функции, които се наричат М-файлове, които решават определен набор от проблеми. Има различни области, в които са налични кутии с инструменти, като цифрова обработка на сигнали, системи за управление, невронна мрежа, симулации, дълбоко обучение и много други области.

R е популярен и мощен език за програмиране с отворен код за статистически изчисления и графики. R прилага различни статистически техники като линейно и нелинейно моделиране, алгоритми за машинно обучение, анализ на времеви редове и класически статистически тестове и т.н. R се състои от език и среда за изпълнение с графики, средство за отстраняване на грешки, достъп до

определени системни функции и възможност за стартиране на програми, съхранени в скриптови файлове.

R е известен със своята стръмна крива на обучение. R е разработен от статистици, следователно пълната му способност е достъпна чрез програмиране. Няма графичен интерфейс, който да помогне на непрограмистите да направят анализа. Работните примери на R са сложни и не са за начинаещи. Въпреки това, R-Commander и R-Studio, новите GUI версии за R, са от полза за общността на разработчиците.

MATLAB, от друга страна, е език, който е лесен за научаване и запомняне, тъй като синтаксисът е прост и последователен по дизайн в продуктите. Що се отнася до технически изчислителни задачи, статистиката и машинното обучение MATLAB е по-бърз от R. Въпреки това опитен разработчик в R може да постигне резултати по-бързо и да подобри производителността.

R и MATLAB са еднакво мощни за визуализиране на данни и извеждане на изходи. R има четири интересни и различни графични реализации - Базова графика, Решетъчна графика, Мрежеста графика и Ggplot2. Базовата графика е графичната система по подразбиране в R и е най-лесната от четирите системи, които да се научат да използват.

MATLAB също поддържа разработване на приложения с функции за графичен потребителски интерфейс (GUI). Графичните функции в MATLAB включват 2D и 3D графични функции за персонализиране на графики интерактивно или програмно. Simulink, допълнителен пакет в MATLAB, е среда за графично програмиране за моделиране, симулиране и анализ на многодомейни динамични системи. Основният интерфейс на Simulink е инструмент за графично блок-схемиране и персонализиран набор от блокови библиотеки. [\[12\]](#)

2. Excel и SPSS

SPSS е съкращение от Statistical packages for Social Science. Той е лидер на пазара по отношение на статистическите инструменти. Има няколко приложения на SPSS, които се считат за производни за манипулиране и съхранение на данни. Той има два метода за пакетна обработка, интерактивни пакети и неинтерактивни пакети.

Excel е един от най-мощните и лесни за използване софтуер за статистика. Той ви позволява да съхранявате данните в табличен формат, във формат на редове и колони. Той също така ви позволява да взаимодействате с вашите данни по

различни начини. Excel има различни функции, които могат да ви помогнат със статистиката. Има няколко начина за импортиране и експортиране на данни. Можете също да интегрирате данните в работния процес. Excel ви позволява да създавате персонализирани функции, използвайки нейните възможности за програмиране.

Основната цел на Excel е да създава записи на данни и да манипулира данните според изискванията на потребителите. Както бе споменато по-рано, Excel ви позволява да използвате външната база данни за анализ, правене на отчети и други. Днес Excel предлага един от най-добрите графични потребителски интерфейси заедно с използването на графични инструменти и техники за визуализация. [\[12\]](#)

2.6 Изводи

След преглед и обстоен анализ между различните съществуващи решения на пазара става ясно, че няма добре унифициран продукт за пълна интеграция със система предоставяща данни, извлечени от видео игри за обучение. Част от софтуерните продукти предоставят достатъчно функционалности за обработка, анализ и визуализация на резултатите, но липсата на интеграция забавя цялостния процес. При нея програмното средство трябва да познава произхода, типа и семантиката на данните получени като игрови метрики и автоматизирано да предоставя резултати от анализи, чрез лесен за използване графичен интерфейс.

Може да се заключи, че дори при наличието на голям брой софтуерни продукти, остава нуждата от софтуерно решение, което да обедини всички изисквания на потребителите и да предостави пълна интеграция със системата за управление.

Глава 3. Използвани технологии и платформи

За реализирането на всяко едно приложение са нужни определен брой средства. Трябва да се определи какво ще бъде нужно, за да се изгради цялостната система за текущото приложение. За да се направи възможно най-добрият избор ще бъдат разгледани и сравнени различни технологии за реализиране на софтуерния продукт.

3.1. Изисквания към средствата

За създаването на уеб-базирано приложение за анализ и визуализация на резултати от играни видео игри за обучение, което да бъде достъпно от всички устройства поддържащи уеб браузър са необходими следните средства:

1. Компютър за разработка на приложението
2. Среда за разработка
3. Езици за програмиране и технологични рамки
4. База данни
5. Хостинг сървър

Компютър за разработка на приложението

Апаратната част (препоръчителен хардуер), която е нужна за разработване на даденото по задание приложение, е следната:

- CPU: Intel Core i7-8750H
- Video Card: NVidia GeForce GTX 1050 Ti
- RAM: 8 GB
- SSD: Samsung EVO 970 240 GB
- HDD: 1000 GB
- Server: XAMPP
- OS: Windows 10 Pro
- Network Adapter: Qualcomm Atheros QCA9377
- Клавиатура: вградена
- Мишка: Logitech G102
- Монитор: Full HD 1920 x 1080

Останалите технологичните средства трябва да се изберат спрямо няколко важни фактора, които ще бъдат разгледани в следващите точки.

3.2. Видове технологии, платформи и място за използването им- сравнителен анализ

За реализиране на приложението ще бъдат разгледани обстойно възможните технологични средства, които покриват необходимите изисквания. Ще се представят техните специфики и на тяхна база ще се направи сравнителен анализ.

Среда за разработка

Интегрираната среда за разработка е софтуерно приложение, което предоставя цялостна среда за разработване на софтуер на програмистите. Тя позволява на разработчиците да започнат бързо да разработват на нови приложения, тъй като не е необходимо множество помощни програми да бъдат ръчно конфигурирани и интегрирани като част от процеса на настройка. Разработчиците също не трябва да прекарват часове, учейки се как да използват различни инструменти, когато всяка помощна програма е представена в една и съща работна среда. Това може да бъде особено полезно при включване на нови разработчици към екипа и по този начин могат да ускорят работния процес. Всъщност, повечето функции на интегрираната среда за разработка имат за цел да спестят време, като интелигентно попълване на код и автоматизирано генериране на код, което премахва необходимостта от въвеждане на пълни символни последователности.

PhpStorm

JetBrains PhpStorm е търговска, междуплатформена IDE (интегрирана среда за разработка) за PHP, изградена на платформата IntelliJ IDEA на JetBrains.

PhpStorm осигурява редактор за PHP, HTML и JavaScript с анализ на кода, предотвратяване на грешки и автоматизирани рефактори за PHP и JavaScript код. PhpStorm поддържа PHP 5.3, 5.4, 5.5, 5.6, 7.0, 7.1 и 7.2 (модерни и стари проекти), включително generators, coroutines, the finally keyword, list in foreach, namespaces, closures, traits and short array syntax. Тя включва SQL редактор с възможност за редактиране на заявки.

PhpStorm е изграден на IntelliJ IDEA, който е написан на Java. Потребителите могат да разширят IDE, като инсталират приставки, създадени за платформата IntelliJ или пишат собствени приставки.

Всички функции, налични в WebStorm, са включени в PhpStorm, което добавя поддръжка за PHP и бази данни. WebStorm се доставя с предварително инсталирани приставки за JavaScript (като например за Node.js). [\[13\]](#)

Microsoft Visual Studio

Microsoft Visual Studio е мощна интегрирана среда за разработка на софтуерни приложения за Windows и за платформата .NET Framework. Използва се за разработка на конзолни и графични потребителски интерфейс приложения, както и Windows Forms или WPF приложения, уеб сайтове, уеб приложения и уеб услуги на всички поддържани платформи от Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework и Microsoft Silverlight.

Visual Studio предоставя мощна интегрирана среда за писане на код, компилиране, изпълнение, дебъгване (както за високо така и за машинно ниво), тестване на приложения, дизайн на потребителски интерфейс (форми, диалози, уеб страници, визуални контроли и други), моделиране на данни, моделиране на класове, изпълнение на тестове, пакетиране на приложения и стотици други функции. Могат да се добавят и плъгини, които повишават функционалността на почти всяко ниво – включително добавянето на поддръжка за source-control системи (като Subversion и Visual SourceSafe), добавяне на нови инструменти като редактори и визуални дизайнери за domain-specific languages или инструменти за други аспекти (като например: Team Foundation Server, Team Explorer). [\[14\]](#)

Eclipse

Eclipse е многоезична среда за разработване на софтуер, която включва интегрирана среда за разработка (IDE) и плъгин система. Написана е главно на Java и може да бъде използвана за разработване на приложения на Java с помощта на различни плъгини и на различни други езици за програмиране, включително Ада, ABAP, C, C++, COBOL, Fortran, Perl, JavaScript, Haskell, PHP, Lasso, Lua, Natural, Python, Ruby, (включително Ruby on Rails framework), Prolog, Clojure, Groovy, Scala, Erlang и Scheme.

Също така може да бъде използван за създаването на пакети за софтуера Mathematica. Средите за разработка включват Eclipse Java Development Tools (JDT) за Java и Scala, Eclipse CDT за C/C++ и Eclipse PDT за PHP, наред с други. Eclipse software development kit (SDK), включва инструментите за разработка на Java, необходими на Java разработчиците. Потребителите могат да разширят възможностите си като инсталират плъгини, написани за платформата на Eclipse, като инструменти за разработка на други програмни езици също така могат да напишат свой плъгин модули.

Разпространяващ се под условията на Eclipse Public License, Eclipse SDK е свободен софтуер с отворен код. Един от първите IDEs работещи под GNU Classpath и също така работи без проблем под IcedTea. [\[15\]](#)

○ **PhpStorm vs Eclipse**

Двете среди за разработка са изключително популярни. PhpStorm е среда фокусирана върху изграждане на приложения с програмния език PHP, предлага всички нужни функции и е по-харесваният избор сред потребителите, спрямо Eclipse. PhpStorm има два вида автоматизирано попълване: структурно попълване и разширяване на думи. Структурното автоматично попълване прави прогнози въз основа на разбирането си за PHP, докато последното се опитва да предвиди думата, която в момента се въвежда, въз основа на предварително въведени думи. Разширяването на думи също работи в коментари и документация и е подобно на многокомплектоването на vim. И двата вида автоматично попълване работят изключително добре, имат малко или никакви проблеми и са доста бързи дори при голямо натоварване.

Eclipse от своя страна има голяма и активна общност, което довежда до голямо разнообразие от приставки (plugins). Eclipse използва персонализиран компилатор (който може да се използва и извън Eclipse), който често е по-бърз от нормалния Java компилатор, особено за постепенно компилиране. Една от най-големите предимства на Eclipse е, че е с отворен код и е безплатна за потребителите.

PhpStorm е платена среда, но поддържа много повече технологични рамки (Symfony, Laravel и други), както и плъгини налични от JetBrains, част от които са Docker, Vagrant, Angular, Vue.js. PhpStorm поддържа както PHP, така и HTML, JavaScript, SQL и други. PhpStorm предлага проверка и автоматично попълване на таблици и полета, както и синхронизиране на написаният код със структурата на базата данни. Един от недостатъците на PhpStorm е необходимите ресурси за да работи бързо и безпроблемно. Средата изисква минимум 2GB RAM, но за предпочитане е да е поне 8GB и четириядрен процесор. За разлика от PhpStorm, Eclipse е подходяща среда за разработка от програмисти, които използват голям набор от програмни езици и търсят безплатна, но професионална и добре поддържана среда. Тя е една от най-бързите среди за разработка на Java приложения и със сигурност би задоволила почти всички изисквания на потребителите. [\[16\]](#)

Езици за програмиране и технологични рамки

Програмирането е усъвършенстване на наученото до момента и непрестанно търсене на нови знания. Накратко представлява писмени инструкции подредени по логичен начин, които компютърът разбира. Дават се малки стъпки от инструкции и компютърът ги изпълнява. Уеб приложенията са програми, които се достъпват чрез уеб браузър.

Уеб приложенията работят на сървър и се достъпват от различни потребители чрез уеб браузъри. Много голяма част от съвременните приложения работят под формата на уеб приложения или мобилни приложения. Някои примери за популярни уеб приложения включват Flickr, Gmail и Google Карти. Част от функционалностите, които уеб приложенията предлагат са, качване и преглеждане на снимки, изпращане на имейли и други.

Уеб програмирането се извършва с езици за уеб програмиране. Тези езици могат да включват статични технологии като HTML, XHTML, CSS, JavaScript и XML. Въпреки това, повечето уеб приложения се програмират с помощта на уеб сървърни езици за програмиране. Този код се изпълнява на сървъра и след това дава статична информация обратно на уеб браузъра. Най-популярните езици за уеб програмиране са: PHP, C# (ASP.NET, ASP classic, .NET), Ruby on Rails, Perl, Python, JavaScript. Тук са представени част от най-известните езици за програмиране на сървърна част (backend):

PHP

PHP е скриптов език върху сървърната (обслужваща) страна език с отворен код, който е проектиран за уеб програмиране и е широко използван за създаване на сървърни приложения и динамично уеб-съдържание. Автор на езика е канадецът от датски произход Размус Лердорф. PHP е рекурсивен акроним от PHP: Hypertext Preprocessor (като в самото начало има значение, дадено от създателите му, на Personal Home Page). Той е един от най-популярните езици за програмиране в Интернет. [\[17\]](#)

PHP се разпространява под отворен лиценз (PHP License), който по своята същност е BSD лиценза и който позволява безплатно разпространяване на програмния код на интерпретатора на езика, както и създаването на производни интерпретатори под други лицензи с уговорката, че тези интерпретатори не могат да включват PHP в името си. Фактът, че PHP се разпространява свободно, го

прави удачен избор за изграждане на Web-сървър базиран изцяло на свободни продукти – GNU/Linux, Apache, MySQL/PostgreSQL и други.

PHP действа главно като филтър, който взима съдържанието на файл и изпълнява специални PHP инструкции, описани във файла. PHP скриптът има начален и краен таг, между които е разположено съдържанието. Отварящият таг е `<?php`, а затварящия е `?>`. Цялата инструкция трябва да завършва с `;`.

Предимства на езика:

- PHP работи на множество операционни системи (Unix, Linux, Windows, BSD, Mac OS X) и множество уеб сървъри – Apache, lighttpd, IIS
- PHP е лесен за разработване
- PHP е безплатен и се разпространява под лиценза на BSD
- PHP може да бъде лесно модифициран и адаптиран към нуждите на прилагашата го организация
- PHP е създаден и пригоден за разработката на уеб приложения
- PHP не изисква особени умения от разработчици работили на структурни езици – езикът е с прост и интуитивен синтаксис за такива разработчици
- PHP е широко разпространен поради простотата си. Има много програмисти, което води до по-евтино платен персонал във фирмите, по-ниска цена на приложенията за клиентите и още по-голяма използваемост. Поддръжката за PHP разработчици е гарантирана от множеството форуми и приложения на общността.
- По аналогия с Perl към стандартните класове на PHP могат да бъдат писани и много допълнителни модули
- PHP поддържа следните системи за управление на бази от данни:
- IBM DB2 – formix – Ingres, Microsoft SQL Server (MS SQL), mSQL, MySQL, Oracle, PostgreSQL, Sybase
- PHP поддържа и ODBC

C#

C# е обектно-ориентиран език за програмиране, разработен от Microsoft, като част от софтуерната платформа .NET. Стремешът още при създаването на C# езика е бил да се създаде един прост, модерен, обектно-ориентиран език с общо предназначение. Основа за C# са C++, Java и донякъде езици като Delphi, VB.NET и C. Той е проектиран да балансира мощност (C++) с възможност за бързо разработване (Visual Basic и Java). Те представляват съвкупност от дефиниции на класове, които съдържат в себе си методи, а в методите е разположена програмната логика – инструкциите, които компютърът изпълнява. Програмите на C# представляват един или няколко файла с разширение .cs., в които се

съдържат дефиниции на класове и други типове. Тези файлове се компилират от компилатора на C# (csc) до изпълним код и в резултат се получават асемблита – файлове със същото име, но с различно разширение (.exe или .dll).

ASP.NET е технология за създаване на уеб приложения и уеб услуги, разработена от „Майкрософт“. За първи път е публикуван през януари 2002 г. с версия 1.0 на .NET Framework, и е наследник на Microsoft Active Server Pages (ASP) технологията, но не е подобрена версия на ASP. ASP.NET е изградена въз основа на Common Language Runtime (CLR), което позволява на програмистите да пишат ASP.NET код като използват .NET език по избор.

Предимства на езика:

ASP.NET цели производителност спрямо останалите скрипт-базирани технологии (включително класическия ASP), като компилира сървърно кода в един или повече DLL файлове на Уеб сървър. Тази компилация става автоматично, когато страницата бива заредена за пръв път (което от своя страна означава, че програмистът не трябва да изпълнява отделни компилации за страниците). Това комбинира лекотата на разработване, предлагана от скриптовите езици, с производителността на бинарните операции. Трябва да се има предвид обаче, че самата компилация може да причини забележимо забавяне при потребителя, когато редактираната страница бива изискана за пръв път от Уеб сървър, но това забавяне не би се появило отново преди следваща промяна.

[\[18\]](#)

Python

Python е интерпретируем, интерактивен, обектно-ориентиран език за програмиране, създаден от Гуидо ван Росум в началото на 90-те години. Кръстен е на телевизионното шоу на BBC „Monty Python’s Flying Circus“. Често бива сравняван с Tcl, Perl, Scheme, Java и Ruby.

Идеята за Python се заражда в края на 1980-те, като реалното осъществяване започва през декември 1989 г. от Гуидо ван Росум в CWI (Centrum Wiskunde & Informatica – международно признат изследователски институт по математика и компютърни науки, базиран в Амстердам, Холандия). Python имал за цел да се превърне в наследник на ABC (език за програмиране, от своя страна вдъхновен от SETL), който да бъде способен да обработва изключения и да е съвместим с операционната система Amoeba. Ван Росум е основният автор на Python, а неговата продължаваща централна роля в развитието на езика е ясно отразена в титлата, дадена му от Python общността – „пожизнен доброжелателен диктатор“ (benevolent dictator for life (BDFL)).

Предимства на езика:

Програмите, написани на Python, са доста компактни и четими, като често те са и по-кратки от еквивалентните им, написани на C/C++. Това е така, тъй като:

- наличните сложни типове данни позволяват изразяването на сложни действия с един-единствен оператор;
- групирането на изразите се извършва чрез отстъп, вместо чрез начални и крайни скоби или някакви други ключови думи (друг език, използващ такъв начин на подредба, е Haskell);
- не са необходими декларации на променливи или аргументи.
- Python съдържа прости конструкции, характерни за функционалния стил на програмиране, които му придават допълнителна гъвкавост.

Всеки модул на Python се компилира преди изпълнение до код за съответната виртуална машина. Този код се записва за повторна употреба като .рус файл. [\[19\]](#)

○ PHP vs C# (ASP.NET)

PHP или Hypertext Preprocessor е високо ниво, динамичен и скриптов език. Той се използва широко при разработването на уеб страници и уеб услугите от страна на сървъра. Добре известен скриптов език от страна на сървъра, PHP често се използва вместо Python.

PHP е свидетел на изоставане от самото начало, тъй като не използва „система за управляван код“, поради което се счита за по -малко сигурна, но позволява на разработчиците да бъдат по -креативни, докато пишат програми.

Използвайки PHP, разработчиците могат да проектират уеб елементи по уникален начин, който не е продиктуван от системата за управляван код от семейството .NET. Освен това PHP е гъвкав поради това, че е с отворен код. Може да работи на всяка платформа.

Предимства на езикът PHP:

1. Събиране на информация за потребителя с помощта на формуляри
2. Събиране на информация автоматично от различни уеб страници и от Интернет
3. Създаване на опция за членство за уебсайт
4. Генериране на динамични файлове и уеб страници
5. Проследяване на потребители с помощта на бисквитки
6. Добавяне на всякакъв вид динамична функция
7. Създаване на настолни приложения
8. Изключително полезни функции за обработка на текст
9. Невероятно лесно намиране на хостинг
10. Широка гама от мощни вградени функции

ASP.NET е рамка за уеб разработка от Microsoft и обикновено използва C#. Рамката изисква разработчиците да работят по определени правила, като например рамката за управляван код, която управлява времето за изпълнение на кода.

Когато използват ASP.NET, разработчиците не се радват на голяма свобода, тъй като трябва да напишат приложение с „базиран на правила“ характер на .NET. Но им позволява да използват модули, които понякога са принудени от системата за управляван код, да работят заедно. Това от своя страна прави разработката, тестването и внедряването на приложенията много по -безпроблемни и по -бързи. Тъй като разработването на надеждни сайтове, използващи системата ASP.NET, не изисква обширен код от нулата - и ASP.NET може да бъде интегриран с мощни системи на Microsoft, като Microsoft SharePoint за разработка на приложения, управление и внедряване навсякъде - уеб разработка с рамката често е по-малко ресурсно натоварена.

Както беше отбелязано, ASP.NET Core, известен преди като ASP.NET VNext, позволява и крос-платформено развитие, което смекчава скъпите лицензи за Windows, необходими за .NET Framework и Microsoft сървъри. [\[20\]](#)

Предимства на езикът C#:

1. Поддържа гъвкаво програмиране поради носенето на обектно-ориентирани функции
2. Поддържа „отделяне на загриженост“, като следва MVC в разработката на приложения. Той също така опростява поддръжката на приложението
3. Насърчава разработката, базирана на тестове, като по този начин единичното тестване позволява на разработчиците да бъдат по-продуктивни
4. Има добре разработена рамка поради наличието на базирани на задачи библиотеки. Независимо дали се изисква редактиране на изображение или XML, разработчиците споделят достатъчно време при разработката на неща.
5. Неговият характер на програмиране, базиран на събития, насърчава простия процес на разработка на приложения
6. Поддържа много различни езици като C#, C, C ++, Jscript.Net, VB. Net и много други. Разработчиците могат да изберат да пишат код в предпочитаните за тях стил

Лесно е да се сравнява един език с друг, но дебатът никога не свършва, докато компаниите не намерят отговора на „какво да изберем за разработването на приложения?“ Освен това ASP.Net и PHP не са забелязали значителен спад в популярността и приемането. PHP, с ръст от 26,4%, е придобил 8 -ма позиция като един от най -популярните езици за програмиране в Stackoverflow Developers Survey 2019.

Табл. 2: Сравнение между C# (ASP.NET) и PHP

	ASP.NET	PHP
Тип	Рамка за уеб приложения и продукт на Microsoft	Скриптов език от страна на сървъра. Използва се от Google, Facebook, Yahoo и др
Поддръжка	Идеален за изграждане на средни до големи корпоративни приложения	Широко използван за създаване на малки до средни уеб решения
Решения	Основен акцент върху сигурността и функционалностите	По-фокусиран върху клиентски, потребителски интерфейси
Общност	Специализирана общност с нарастващи разработчици	Общност с голям размер
Сигурност	Много сигурен	По-малко вградена функция за сигурност
Бързина	Прилична скорост, достатъчно бърза за настолно приложение	Не е подходящо и по -бавно за настолно приложение
Персонализиране	По -малко склонни към персонализиране	Позволява персонализиране, което може да причини грешки, което води до уязвимост към лошо кодиране спрямо .NET

- **PHP vs Python**

Python е интерпретиран и обектно-ориентиран език за програмиране на високо ниво, който предлага огромна библиотечна поддръжка и се използва за

разработване на самостоятелни програми и скриптов алгоритми за различни области. Той е създаден от Guido Van Rossum и издава първата си версия през 1990 г. [21]

Хипертекстовият препроцесор, изобретен през 1995 г., известен също като PHP, е сървърно-скриптов език. Използва се за създаване на динамично HTML съдържание в мрежата. Популярно се използва за генериране на XML документи, Flash анимации, графики, PDF файлове и много други.

- Python е мощен, преносим, с отворен код и е сравнително лесен за научаване и забавен за използване. Той има много функции, които други езици за програмиране не поддържат. Синтаксисът му е по -прост и кодът е по -четим в Python в сравнение с други езици за програмиране като PHP, C и C ++.
- PHP не се използва за програмиране с общо предназначение и се използва само за създаване на динамично уеб съдържание с html. Единствената причина да се придържате към PHP е лесната му употреба.

Табл. 3: Сравнение между Python и PHP

	Python	PHP
Крива на обучение	Python е по -добър от PHP в дългосрочен проект	PHP има ниска крива на обучение, лесно е да започнете с PHP
Рамка	В сравнение с PHP, Python има по -малък брой рамки. Популярни са Django, Flask	PHP има огромен брой рамки. Популярни са Laravel, Slim
Синтаксис	Синтаксисът е лесен за запомняне, почти подобен на човешкия език	Синтаксисът е малко необичаен в сравнение с Python, той има широк спектър от договорености за именуване
Най – важни функционалности	По -малко ред код, бързо разгръщане и динамично въвеждане	Отворен код и лесно внедряване
Тип език	Това е език за програмиране с общо предназначение	Това е език за програмиране за уеб разработка

Среда в която е популярен	Задача за машинно обучение, наука за данни, изкуствен интелект и автоматизация	Избор на език в уеб разработката
Поддръжка	В сравнение с PHP той е по -лесен за поддръжка	Малко труден за поддръжка
Темпо на популярност	След 2016 г. популярността на Python се увеличава бързо	PHP губи популярността си в Stack Overflow

Тук са представени част от най-известните технологии и езици за програмиране на потребителската част (frontend):

HTML

(HyperText Markup Language) е най -основният градивен елемент в мрежата. Той определя смисъла и структурата на уеб съдържанието. Други технологии освен HTML обикновено се използват за описание на външния вид/презентацията на уеб страница (CSS) или функционалността/поведението (JavaScript).

"Хипертекст" се отнася до връзки, които свързват уеб страници една с друга, или в рамките на един уебсайт, или между уебсайтове. Връзките са основен аспект на мрежата.

HTML използва "маркиране", за да коментира текст, изображения и друго съдържание за показване в уеб браузър. HTML маркирането включва специални „елементи“ като <head>, <title>, <body>, <header>, <footer>, <article>, <section>, <p>, <div>, , , <отстрани>, <audio>, <canvas>, <datalist>, <details>, <embed>, <nav>, <output>, <progress>, <video>, , , и много други.

HTML елемент се отделя от друг текст в документ чрез "тагове", които се състоят от името на елемента, заобиколен от "<" и ">". Името на елемент вътре в етикета не е чувствително към регистъра. Тоест може да се пише с главни, малки или смес. Например маркерът <title> може да бъде написан като <Title>, <TITLE> или по друг начин. [\[22\]](#)

CSS

CSS (Cascading Style Sheets) е език за описване на презентацията и стиловете на елементите в един HTML/XML документ. CSS е една от основните технологии, използвани в Уеб, редом с HTML и JavaScript.

При създаване на HTML страница, съдържанието ѝ се описва (маркира) с HTML код, а презентацията на това съдържание, тоест как ще изглежда то в браузъра, се описва с CSS код (стил).

CSS кодът се поставя във файл с разширение .css. След това този файл може да се използва във всяка една страница от уебсайта. Така че стилът ще се намира на едно място и при промяна ще се отразява на всички уеб страници.

Съществуват няколко начина за прилагане на CSS стила:

- **Външен стил**

Използва се когато трябва да се контролират множество HTML документи, като нужните параметри се задават във външен файл (**file.css**).

- **Вътрешен стил**

Използва се за да се зададе вида на един отделен HTML документ, като нужните свойства се задават със специалния таг **<style>** в секцията **<head>** на HTML страницата.

- **Вграден стил**

CSS стилът се разполага като атрибут директно в HTML тага например **<p>**.

Когато в един документ се използват и трите начина за налагане на CSS стил – с най-висок приоритет е стилът на Inline Style (вътрешните за HTML таговете стилове), след тях по приоритет са Internal Style Sheet (стиловете от секцията head на HTML документа) и последни по приоритет са External Style Sheet, т.е. стиловете, декларирани във външен CSS файл. [\[23\]](#)

JavaScript

JavaScript (JS) е лек, интерпретиран или точно навреме компилиран език за програмиране с първокласни функции. Въпреки че е най-известен като скриптовия език за уеб страници, много среди, които не са браузъри, също го използват, като Node.js, Apache CouchDB и Adobe Acrobat. JavaScript е прототип, базиран на много парадигми, еднопоточен, динамичен език, поддържащ обектно-ориентирани, императивни и декларативни (напр. Функционално програмиране) стилове.

Не трябва да се бърка JavaScript с езика за програмиране Java. Както „Java“, така и „JavaScript“ са търговски марки или регистрирани търговски марки на Oracle в

САЩ и други страни. Двата езика за програмиране обаче имат много различен синтаксис, семантика и употреба. [\[24\]](#)

- **TypeScript**

По дефиниция „TypeScript е JavaScript за разработка на мащабни приложения“. TypeScript е силно типизиран, обектно ориентиран, компилиран език. Проектиран е от Андерс Хейлсберг (дизайнер на C#) в Microsoft. TypeScript е както език, така и набор от инструменти. TypeScript е типизирано супермножество JavaScript, компилирано в JavaScript. С други думи, TypeScript е JavaScript плюс някои допълнителни функции.

Характеристики на TypeScript

TypeScript е просто JavaScript. TypeScript започва с JavaScript и завършва с JavaScript.

TypeScript поддържа други JS библиотеки. Компилираният TypeScript може да се използва от всеки JavaScript код. JavaScript, генериран от TypeScript, може да използва повторно всички съществуващи JavaScript рамки, инструменти и библиотеки.

JavaScript е TypeScript. Това означава, че всеки валиден .js файл може да бъде преименуван на .ts и компилиран с други TypeScript файлове.

TypeScript е преносим в браузъри, устройства и операционни системи. Той може да работи във всяка среда, в която работи JavaScript. За разлика от своите колеги, TypeScript не се нуждае от специализирана виртуална машина или специфична среда за изпълнение, за да се изпълни. [\[25\]](#)

Тук са представени различни видове технологични рамки за разработка на уеб приложения:

Laravel

Laravel е рамка за уеб приложения с изразителен, елегантен синтаксис. Уеб рамката предоставя структура и отправна точка за създаване на приложения.

Laravel се стреми да осигури невероятно изживяване на разработчиците, като същевременно предоставя мощни функции, като цялостно инжектиране на зависимости, експресивен слой за абстракция на база данни, опашки и планирани задачи, тестване на модули и интеграция и други.

Има голямо разнообразие от инструменти и рамки, които са достъпни при изграждането на уеб приложение. Laravel е добър избор за изграждане на съвременни уеб приложения с пълен стек.

Laravel е „прогресивна“ рамка. Има огромна библиотека с документация, ръководства и видео уроци на Laravel. Laravel предоставя здрави инструменти за инжектиране на зависимости, единично тестване, опашки, събития в реално време и други. Laravel е добре настроен за създаване на професионални уеб приложения и е готов да се справи с големи натоварвания.

Laravel е невероятно мащабируем. Благодарение на естеството на мащабиране на PHP и вградената поддръжка на Laravel за бързи, разпределени кеш системи като Redis, хоризонталното мащабиране с Laravel е лесно. Всъщност приложенията на Laravel са лесно мащабирани, за да обработват стотици милиони заявки на месец. Платформи, като Laravel Vapor позволява да се стартира приложение на Laravel в почти неограничен мащаб с най -новата безсървърна технология на AWS. Laravel съчетава най -добрите пакети в PHP екосистемата, за да предложи най -здравата налична рамка, подходяща за разработчиците. В допълнение, хиляди талантиливи разработчици от цял свят са допринесли за рамката. [\[26\]](#)

Symfony

Symfony е PHP рамка с отворен код за уеб приложения и набор от PHP компоненти за многократна употреба. Първоначално е замислен през 2005 г. от интерактивната агенция SensioLabs за разработване на уеб сайтове за собствени клиенти.

Хиляди уеб сайтове и приложения разчитат на Symfony като основа на своите уеб услуги. И повечето от водещите PHP проекти, като Drupal и Laravel използват компоненти Symfony за изграждане на своите приложения.

Уеб рамката на PHP е колекция от класове, която помага за разработването на уеб приложение. Symfony е MVC рамка с отворен код за бързо развиващи се съвременни уеб приложения. Symfony е пълнофункционална уеб рамка. Той съдържа набор от PHP компоненти за многократна употреба. Можете да използвате всякакви компоненти на Symfony в приложения, независимо от рамката.

Symfony има огромно количество функционалност и активна общност. Той има гъвкава конфигурация, използваща YAML, XML или пояснения. Symfony се интегрира с независима библиотека и PHP единица. Symfony е вдъхновен предимно от Ruby on Rails, Django и Spring рамки за уеб приложения. Компонентите на Symfony се използват от много проекти с отворен код, които включват Composer, Drupal и phpBB.

Рамката Symfony се състои от няколко компонента, като например компонента HttpFoundation, който разбира HTTP и предлага хубав обект на заявка и отговор, използван от другите компоненти. Други са само помощни компоненти, като Validator, които помагат за валидиране на данни. Компонентът на ядрото е сърцето на системата. Ядрото е основно „основният клас“, който управлява околната среда и носи отговорността да обработва http заявка.

Добре организираната структура на Symfony, чистият код и добрите програмни практики улесняват уеб разработката. Symfony е много гъвкав, използван за изграждане на микросайтове и обработка на корпоративни приложения с милиарди връзки. [\[27\]](#)

- **Laravel vs Symfony**

Laravel е рамка с отворен код, която следва модела на проектиране на модел-изглед-контролер. Той използва повторно съществуващите компоненти на различни рамки, за да създаде уеб приложение. Той също така се състои от основни функции на PHP рамки като Yii, CodeIgniter или Ruby on Rails. Laravel е добре известен с простото кодиране, което се доближава и намалява времевата рамка за разработка, което е чудесно за разработване на просто PHP приложение.

Symfony-тази рамка също се основава на PHP проекти с отворен код като Propel, Doctrine, PHPUnit, Twig и Swift Mailer. Въпреки факта, че има своите компоненти като Symfony YAML, Symfony Event Dispatcher, Symfony Dependency Injector и Symfony Templating. От 2005 г. Symfony се издига като все по -надеждна и зряла рамка. Използва се главно за сложни корпоративни проекти.

Прилики

Двете рамки използват PHP като език за програмиране.

Те са междуплатформени, което означава, че са компютърен софтуер, който се реализира на множество изчислителни платформи.

Двете осигуряват скелет на приложението, модел за интерфейси и поддържат търсене на текст.

Разлики

Symfony може да бъде наречен като конвенционален PHP език - може да бъде променен на C# или Java, но разбира се, той се състои от уникални и единствени по рода си елементи, които го правят изключителен. През 2021 г. Laravel се появи като най -популярната PHP рамка. Той разчита повече на магически методи и черти. Това прави кода по -кратък и цялата рамка е по -лесна за разбиране.

Symfony е предназначен за малко по-машабни или по-сложни проекти, съдържащи огромни функции и използван от значителен брой клиенти. Laravel е свързан с модела на проектиране на MVC, който е споменат по-горе. Когато става въпрос за мащабируемост, Symfony предоставя няколко платформи за поддържане на мащабируемост.

Symfony доставя Twig Templating engine. Laravel от своя страна предоставя Blade Templating engine, което има голямо предимство - можете да използвате кода за многократна употреба. Тази опция не съществува в Twig. [\[28\]](#)

Бази данни

Бази данни наричаме организирана колекция от данни, обикновено съхранявана и достъпна по електронен път от компютърна система. Ще бъдат представени два типа бази данни:

1. Релационна база данни - Срещани и като „SQL бази данни“, заради езика за запитвания SQL, съхраняват данните по предварително структуриран начин – в таблици, подредени в редове (записи) и колони. Между отделните данни и таблици може да се създават връзки (релации). Таблиците и връзките между тях образуват структура, която може да се представи като схема.

Отделните единици информация се съдържат в единични полета, подредени в редове в таблица. Допълнителна информация за данните се записва чрез колоните, в допълнителни полета към реда/записа. Например записът за статия в WordPress базата данни, в таблицата **posts**, съдържа уникален номер на статията (**ID**), който не се променя. Всяка следваща колона съдържа допълнителна информация, която може да се променя като: съдържание (`post_content`), име на публикация (`post_title`), статус (`post_status`), автор и други. Други данни, свързани с дадена статия, но различни по смисъл, може да се съхраняват в отделни таблици. Например таблицата **comments**, в която се съхраняват коментарите към дадената статия. Връзката (релацията) между двете таблици `posts` и `comments` използва уникалния номер на публикацията. Когато статията и коментарите ѝ трябва да се покажат на една страница в сайта, чрез SQL езика се създава запитване към базата данни. SQL запитването се приема и изпълнява от контролера на базата (`database engine`), който намира и съединява данните от двете таблици и ги връща като резултат.

Схемата на данните е фиксирана и с предварително зададени спецификации. За да може да се добавя информация в релационната база

данни, първо в нея трябва да се дефинира схемата, таблиците, полетата и типа им, ключовете, връзките, тригери и други. Промени на вече създадена база данни са възможни, но когато те са големи, това е сложен процес, който може да наложи и временна недостъпност до данните.

2. Нерелационна база данни - NoSQL базите данни са общо наименование на различни технологии за бази данни, създадени за нуждите на модерните приложения и огромното количество информация, с което те работят. С NoSQL базите данни се решават различни SQL ограничения за:

- лесна мащабируемост върху клъстери от сървъри (хоризонтално мащабиране)
- поддръжка на различни типове структури от данни;
- използване при разработка с гъвкави методологии (agile development)
- Някои NoSQL бази данни може да не отговарят напълно на ACID (Atomicity, Consistency, Isolation, Durability) модела за транзакции. ACID моделът е колекция от свойства на транзакциите, чрез които може да се гарантира цялост, завършеност, изолация и устойчивост на транзакциите в базата данни
- Някои NoSQL бази също така може да не поддържат join операциите, използвани в релационните бази данни. Вместо тях, за обработката на свързани данни, се използват различни подходи като заместители: няколко заявки, вместо една; кеширане, репликиране, де-нормализиране и вместване (nesting) на данни

○ MySQL vs MongoDB

MySQL е популярна, безплатна за използване и с отворен код система за управление на релационни бази данни (RDBMS), разработена от Oracle. Както при другите релационни системи, MySQL съхранява данни, използвайки таблици и редове, налага референтна цялост и използва структуриран език за заявки (SQL) за достъп до данни. Когато потребителите трябва да извличат данни от база данни MySQL, те трябва да конструират SQL заявка, която да обединява множество таблици заедно, за да създаде изгледа на данните, които им трябва.

Схемите на базата данни и моделите на данни трябва да бъдат дефинирани предварително и данните трябва да съответстват на тази схема, за да се съхраняват в базата данни. Този строг подход към съхранението на данни предлага известна степен на безопасност, но го променя за гъвкавост. Ако трябва да се съхранява нов тип или формат на данни в базата данни, трябва да се извърши

миграция на схема, която може да стане сложна и скъпа с увеличаването на размера на базата данни.

MongoDB също е безплатен за използване и с отворен код. Принципите му на проектиране обаче се различават от традиционните релационни системи. Често оформена като нерелационна (или NoSQL) система, MongoDB възприема значително различен подход за съхранение на данни, представяща информацията като поредица от JSON-подобни документи (всъщност съхранявани като двоичен JSON или BSON), за разлика от таблицата и ред формат на релационни системи. Документите MongoDB се състоят от поредица двойки ключ/стойност от различни типове, включително масиви и вложени документи. Ключовата разлика обаче е, че структурата на двойките ключ/стойност в дадена колекция може да варира от документ на документ. Този по-гъвкав подход е възможен, тъй като документите се самоописват.

Основните разлики между тези две системи за бази данни са значителни. Изборът какво да се използва е наистина въпрос на подход, а не на чисто техническо решение.

MySQL е зряла система за релационни бази данни, предлагаща позната среда за бази данни за опитни ИТ специалисти.

MongoDB е утвърдена, нерелационна система от бази данни, предлагаща подобрена гъвкавост и хоризонтална мащабируемост, но с цената на някои функции за безопасност на релационни бази данни, като например референтна цялост.

MongoDB е привлекателна опция за разработчиците. Неговата философия за съхранение на данни е проста и веднага разбираема за всеки с опит в програмирането.

MongoDB съхранява данни в колекции без наложена схема. Този гъвкав подход за съхранение на данни го прави особено подходящ за разработчици, които може да не са експерти по бази данни, но все пак искат да използват база данни за подпомагане на разработването на техните приложения.

В сравнение с MySQL, тази гъвкавост е значително предимство: За да се извлече максимум от релационната база данни, първо трябва да се разберат принципите на нормализиране, референтна цялост и дизайн на релационни бази данни.

С възможност за съхраняване на документи с различни схеми, включително неструктурирани набори от данни, MongoDB предоставя гъвкав интерфейс за разработчици за екипи, които изграждат приложения, които не се нуждаят от всички функции за безопасност, предлагани от релационните системи. Често срещан пример за такова приложение е уеб приложение, което не зависи от

структурирани схеми; може лесно да обслужва неструктурирани, полуструктурирани или структурирани данни, всички от една и съща колекция MongoDB.

MySQL е общ избор за потребители, които имат богат опит в използването на традиционни SQL скриптове, проектират решения за релационни бази данни или които променят или актуализират съществуващи приложения, които вече работят с релационна система. Релационните бази данни също могат да бъдат по -добър избор за приложения, които изискват много сложни, но твърди структури от данни и схеми на бази данни в голям брой таблици.

Често срещан пример за такава система може да бъде банково приложение, което изисква много силна референтна цялост и гаранции за транзакции, за да се поддържа точна целостта на данните в даден момент.

Важно е обаче да се изясни, че MongoDB също поддържа ACID свойства на транзакциите (автономност, последователност, изолация и трайност). Това позволява по-голяма гъвкавост при изграждането на модел на данни за транзакции, който може да се мащабира хоризонтално в разпределена среда и няма влияние върху производителността при транзакции с множество документи.
[\[29\]](#)

3.3. Избор на средствата

След обстоен анализ на наличните технологии за разработка на уеб приложения и след направените сравнителни анализи, бяха избрани следните:

Среда за разработка

PhpStorm – Среда за разработка, разработена предимно за PHP интеграция, дава абсолютно всички нужни функционалности за разработката на дипломната работа. Осигурява редактор за PHP, HTML и JavaScript с анализ на кода, предотвратяване на грешки и автоматизирани рефактори за PHP и JavaScript код. Поради факта, че поддържа много голям набор от PHP рамки и плъгини е избрана за разработката на уеб приложението.

Език за програмиране и технологична рамка

PHP - Фактът, че PHP се разпространява свободно, го прави удачен избор за изграждане на Web-сървър базиран изцяло на свободни продукти – GNU/Linux, Apache, MySQL/PostgreSQL и други. Поради ограниченията от към избора на

сървър, даденото уеб приложение трябва да се разработи на даденият език. Огромна част от всички уеб приложения по света все още използват PHP като език за програмиране, това го прави лесен за поддръжка и намиране на софтуерни разработчици.

Laravel - Laravel съчетава най-добрите пакети в PHP екосистемата, за да предложи най-здравата налична рамка, подходяща за разработчиците. В допълнение, хиляди талантиливи разработчици от цял свят са допринесли за рамката. Тази рамка предоставя сигурност при разработката на приложения, които ще се разрастват с времето, като по този начин гарантиран безотказна работа и лесно разширение на функционалностите на приложението.

База данни

MySQL – Една от най-използваните релационни база данни в света. Тя предлага изключително много възможности за съхранение на данни в структуриран и защитен вид. Когато потребителите трябва да извлекат данни от базата, те използват SQL заявки. Почти всеки програмист разбира от SQL заявки. Лесни и близки до човешкия език, те предлагат бърз начин за менажиране на информацията съхранявана в базата данни. Тя предлага и богат избор от графични софтуери, които предлагат лесна визуализация и менажиране на данните в таблици. [\[30\]](#)

HeidiSQL - HeidiSQL е безплатен софтуер и има за цел да бъде лесен за научаване. "Heidi" позволява да се виждат и редактират данни и структури от компютри, работещи с една от системите за бази данни MariaDB, MySQL, Microsoft SQL, PostgreSQL и SQLite. Изобетен през 2002 г. от Ansgar, HeidiSQL принадлежи към най-популярните инструменти за MariaDB и MySQL по целия свят. С негова помощ, лесно и удобно може да се визуализират данните в базата данни извлечени от видео игрите за обучение.

Сървър

Споделен хостинг – За целта ще се използва услугата за споделен хостинг на SuperHosting. Тя предлага необходимите параметри за работа на уеб приложението. Сървърът има инсталиран PHP и има дисково пространство от 15 GB. Месечно около 12 000 посещения могат да бъдат направени от потребителите. Компанията предлага също денонощна поддръжка и цялостна защита от хакерски атаки.

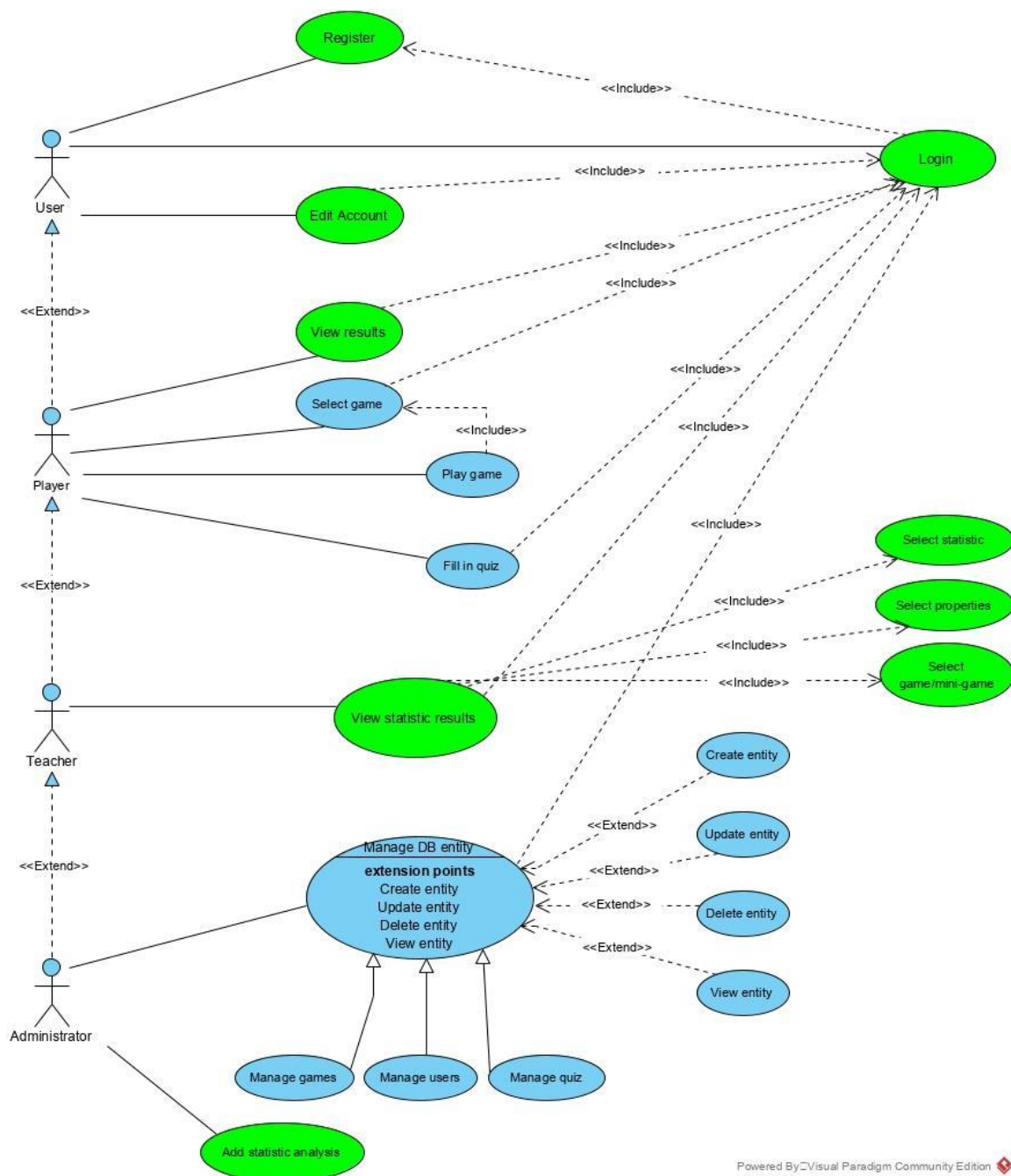
3.4. Изводи

След представените технологии за разработка на уеб приложението и направените сравнителни анализи, бяха избрани най-подходящите при дадените условия средства за реализирането на дипломният проект. Бяха представени техните предимства и недостатъци, както и ограниченията при изборът им. Както стана ясно, уеб приложението може да се реализира бързо и качествено и да отговаря на всички изисквания на продукта.

Глава 4. Анализ

За реализирането на качествен софтуерен продукт, е необходимо в начален етап да се определи каква е идеята на продукта, какви са очакванията на заинтересованите страни към него и да се преминат важни стъпки за осигуряване на неговата сигурност, скалируемост, функционалности, поддръжка и други. За да се изгради успешен софтуер, е нужно да се направи и добър анализ, който ще доведе и до доволни клиенти.

4.1. Концептуален модел



Фиг. 5 Диаграма на случаи на употреба за концептуален анализ

Фиг.5 представя диаграма на случаи на употреба за концептуален анализ на данни. От диаграмата може да се заключи кои са основните актьори (действащи лица) в системата и кои са действията, които всеки един от тях може да извършва. **Със зелен цвят са изобразени всички случаи на употреба (т.е. потребителски действия), които са разработени в настоящата дипломна работа.**

Част от концептуалния модел на системата са следните обекти, или наричани основни действащи актьори са:

- Потребители от тип играч
- Потребители от тип учител

Абстрактният клас Потребител може да извършва началните действия в системата и той е базов клас за всички останали класове. Всички действащи актьори разширяват и наследяват всички действия, които по-горните актьори могат да извършват в системата.

Всички възможни функционалности , в зависимост от ролята на потребителите са:

- **Регистрация на потребител**
Всеки потребител на системата може да се регистрира, чрез въвеждане на име и фамилия, имейл адрес и парола за достъп
- **Вход в системата**
Всеки регистриран потребител, може да влезе в системата, като въведе неговия имейл адрес и парола
- **Промяна на данните на профила на потребителя**
Всеки регистриран и влезнал в системата потребител, има възможност да промени своите данни, като име и фамилия, парола, години, пол и успех от меню „Профил“
- **Попълване на анкети**
Всеки потребител влезнал в системата, може да попълни анкети за определяне на дадено качество на потребителя
- **Избор на игра и стартиране**
Всеки потребител влезнал в системата, може да избере игра и да я стартира
- **Преглед на резултатите от изиграни видео игри за обучение**

Всеки потребител влезнал в системата, може да прегледа резултатите от играните видео игри за обучение на всички потребители от меню „Резултати“

- **Преглед на статистически резултати**

Потребителите с роля *учител* могат да правят статистически анализ върху резултатите от играните видео игри за обучение

- **Добавяне на нова статистика**

Потребителите с роля *администратор* могат да добавят нов вид статистики в системата

4.2. Потребителски изисквания

Потребителските изисквания описват какви функционалности трябва да може да съдържа системата. Това са всички действия, които са видими от потребителският интерфейс.

Всички роли в системата са:

- Играч
- Учител
- Администратор

Действията, които могат да изпълняват всички потребители на системата са:

- Регистриране на профил, чрез въвеждане на име и фамилия, имейл адрес и парола;
- Вход в системата чрез имейл адрес и парола за достъп, които потребителят е въвел при регистрация;
- Възможност за промяна на данните на профила, като име и фамилия, парола, години, пол и успех;

Действията, които могат да се изпълняват от роля *играч*:

- Преглед на резултатите от играни видео игри за обучение на всички потребители в системата;
- Избор на игра за обучение
- Стартиране на игра за обучение

Действията, които могат да се изпълняват от роля *учител*, се дефинират така:

Всички описани действия, които могат да се изпълняват от роля *игрaч*, се изпълняват и от роля *учител*;

- Преглед на резултати от статистически анализ на данни, събрани от играните видео игри за обучение. За да се генерират резултатите от статистическия анализ, трябва да се изпълнят няколко стъпки:
 - да се избере игра / мини-игра
 - да се изберат полета (данни), върху които ще бъде направен статистическият анализ
 - да се избере метод за статистически анализ (Корелация, Т тест, ANOVA, Effect size)

Всеки метод за статистически анализ показва резултат в потребителският интерфейс. Тук е обяснено значението на всеки един символ след генериране на резултатите от статистическия анализ:

- n – брой елементи в извадката
- M – средноаритметична стойност
- SD – стандартно отклонение
- SE – стандартна грешка
- Т-тест
 - $P1$ - one-tailed p value (с една опашка)
 - $P2$ - two-tailed p value (с две опашки)
 - T score – резултат от теста
 - DF – степен на свобода
- ANOVA
 - F – резултат от теста
 - P – p value (p - стойността)

Действията, които могат да се изпълняват от роля *администратор*:

Всички описани действия, които могат да се изпълняват от роля *игрaч* и *учител*, се изпълняват и от роля *администратор*;

- Менажиране на базата данни. Добавяне, обновяване, изтриване и преглед на данните. Данните могат да бъдат видове игри, потребители, анкети;
- Добавяне на нов вид статистически метод в системата;

4.3. Качествени изисквания

Качествените изисквания определят как ще се имплементират функционалните изисквания в системата. Те определят различни характеристики или ограничения

на имплементацията на софтуера, неговата поддръжка и държание в различни ситуации. Ще се разгледат различните характеристики на качеството на софтуера:

- Сигурност
- Изправност
- Изменяемост
- Производителност

Тези характеристики са основополагащи за проектирането, планирането и имплементирането на качествен софтуер.

Сигурност

Всеки потребител се автентикира в системата чрез имейл адрес и парола за достъп. Паролата за достъп трябва да е не по-малка от шест символа. За всеки един потребител автентикирал се в системата се издава случаен токен (API token) за достъп. Той служи за автентикиране на потребителя при извличане на информация от сървъра чрез HTTP заявки.

Всеки потребител на системата има определена роля след регистриране. Ролите в системата са:

- Играл
- Учител
- Администратор

Всяка роля служи за различни предоставени функционалности и нива на достъп до системни ресурси. Оторизацията е процесът, който дава на потребителя разрешение за достъп до определен ресурс или функция. Този термин често се използва взаимозаменяемо с контрол на достъпа или клиентски права.

В защитени среди оторизацията винаги трябва да следва автентикацията. Потребителите първо трябва да докажат, че самоличността им е истинска, преди администраторите да им предоставят достъп до исканите ресурси и функционалности. В системата има ограничения, потребители с роля *играл* не могат да преглеждат резултати от статистически анализ на резултати на потребители от играни видео игри за обучение.

Паролите на потребителите се пазят криптирани в базата данни. При злонамерен опит за открадване на потребителски данни, извършителят не може да използва паролата за достъп на друг потребител.

Изправност

В проекта е имплементирано обработка на изключения на системата. При възникване на грешка си извиква изключение, което ясно и точно може да представи информация за грешката, къде е възникнала и при кой процес. Това прави отстраняването на грешки от софтуера много по-бързо и точно. При

въвеждане на грешни данни от потребителя, системата показва съобщение на екрана с информация за възникналата грешка.

В една система трябва да има определен брой заявки за достъп до даден ресурс. Ако този брой заявки бъде достигнат системата трябва да реагира подобаващо, като блокира последващи опити за достъп и изпрати отговор за неуспешна заявка. По този начин могат да се избегнат сризове в системата или претоварване на ресурсите на сървъра.

При неуспешна заявка към сървъра за записване на данни, цялата извършена дейност до момента се връща в предишното състояние. По този начин се получава консистентност на данните. Сървърът изпраща съобщение за грешка към клиента.

При определени действия от страна на потребителя, се презарежда потребителският интерфейс за да се предостави консистентност на данните.

Изменяемост

Употреба на интерфейси е с цел по-слабо свързване между отделни елементи на системата или така нареченото „Loose coupling“. Разхлабеното свързване е подход за взаимосвързване на компонентите в система или мрежа, така че тези компоненти, наричани още елементи, да зависят един от друг в най-малка степен на практика. Свързването се отнася до степента на преки познания, които един елемент има за друг.

Целта на хлабавата свързана архитектура е да намали риска промяната, направена в един елемент, да създаде неочаквани промени в други елементи.

Производителност

Използването само на нужни данни при извикване на заявка за предоставяне на информация от сървъра. Избира се само тази част от информацията на дадени сложни обекти, която е нужна на потребителя. По този начин натоварването на системата се намалява.

Системата трябва да поддържа висок брой потребители, използващи системата едновременно без да има забавяне от нейна страна. Добър начин за проследяване на производителността на системата, е чрез използване на стрес тестове. Използват се за намиране на грешки, при определяне на надеждността на софтуера чрез тестване извън границите на нормалната работа. Стрес тестването е особено важно за критичен софтуер, но се използва за всички видове софтуер.

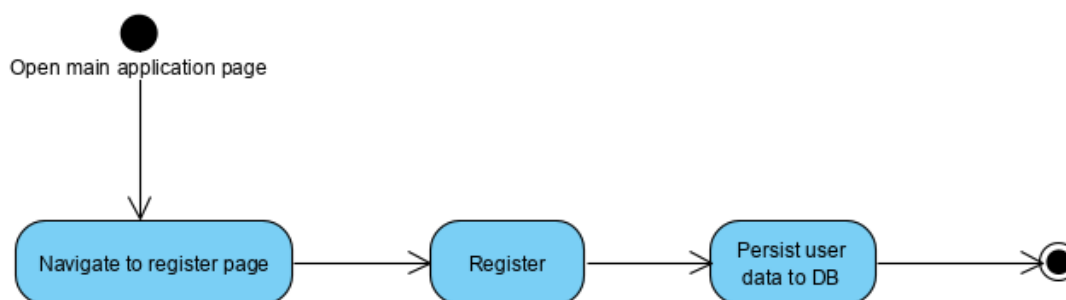
Стрес тестването акцентира върху здравината, наличността и обработката на грешки при голямо натоварване, а не върху правилното поведение при нормални ситуации.

4.4. Работни процеси

Ще бъдат разгледани част от работните процеси на системата, чрез диаграми на дейностите.

- Регистрация

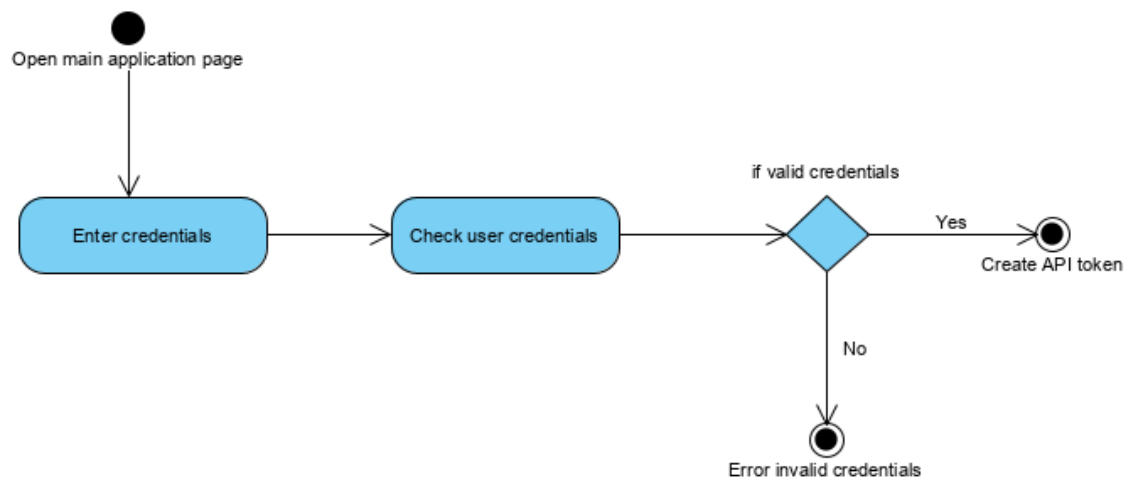
На Фиг. 6 е представена диаграма на регистрация на потребител в системата. При регистрация, потребителя избира опцията „Регистрирай се от тук“ от екран „Вход“. Отваря се страницата за регистрация на потребител, в която трябва да се въведат име и фамилия, имейл адрес и парола за достъп. След което се натиска бутон „Регистрация“. Данните на потребителя се изпращат до сървъра, където те биват проверени за коректност и запазени в базата данни. Паролата се запазва в криптиран вид.



Фиг. 6 Регистрация на потребител в системата

- Вход в системата за регистрирани потребители

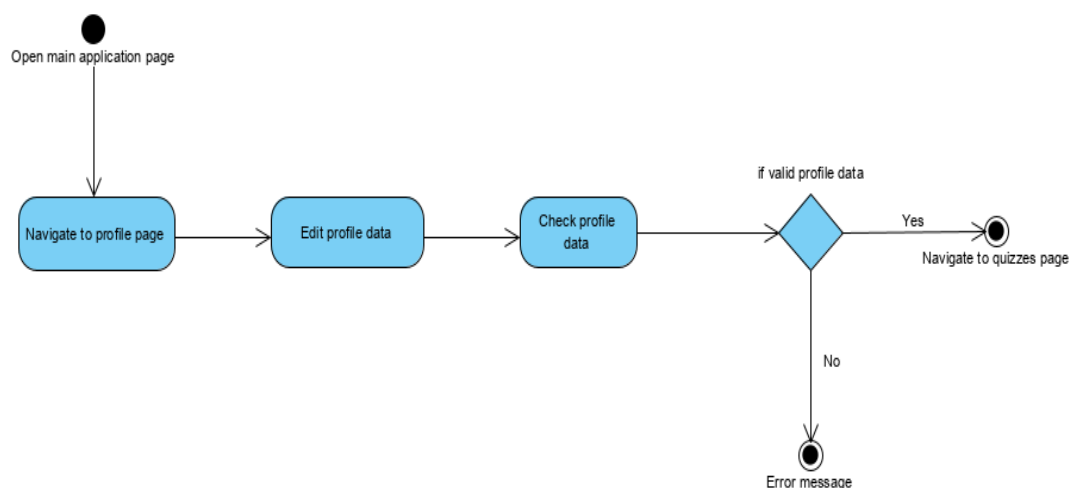
На Фиг. 7 е представена диаграма на вход в системата на регистрирани потребители. След успешна регистрация, потребителят може да влезе в системата с въведените при регистрация имейл адрес и парола. При невалидни данни се изпраща грешка от системата към потребителския интерфейс за грешно въведен имейл или парола. При валидни данни се издава токен, който служи за автентикация и оторизация на потребителя в системата.



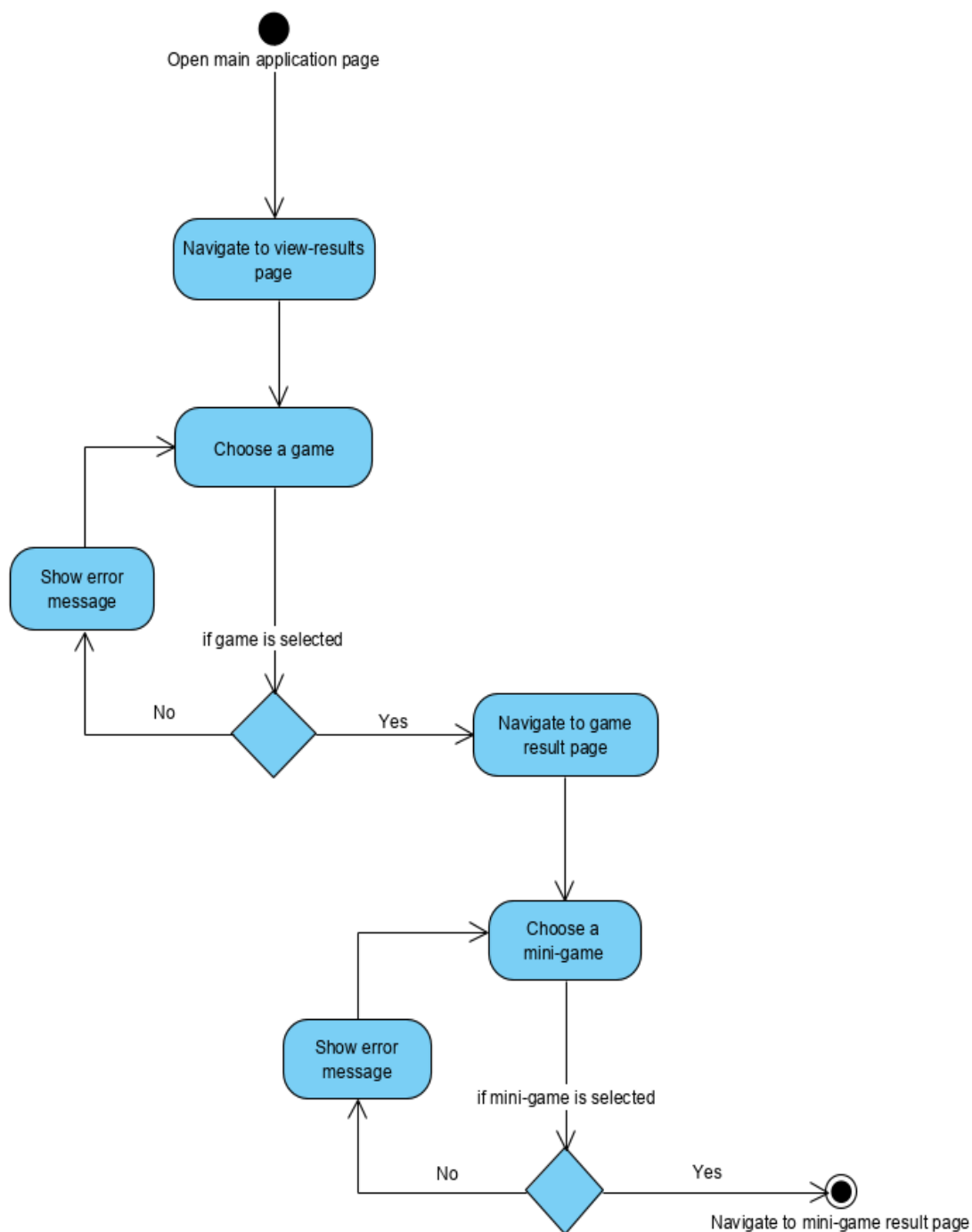
Фиг. 7 Вход на потребител в системата

- Редактиране на данните за профила

На Фиг. 8 е представена диаграма на редактиране на потребителските данни на даден профил. След успешна регистрация и вход в системата потребителят може да промени своите данни чрез навигиране към меню „Профил“. Данните, които могат да се променят са име, фамилия, парола, години, пол и успех. След редактиране на данните се натиска бутон „Запази“. При грешно въведени данни или пропуснати полета от страна на потребителя, системата изпраща съобщение за грешка към потребителския интерфейс. При валидни данни, системата запазва новите промени в базата данни и навигира потребителя до меню „Анкетите“.



Фиг. 8 Редактиране на потребителски данни на профил

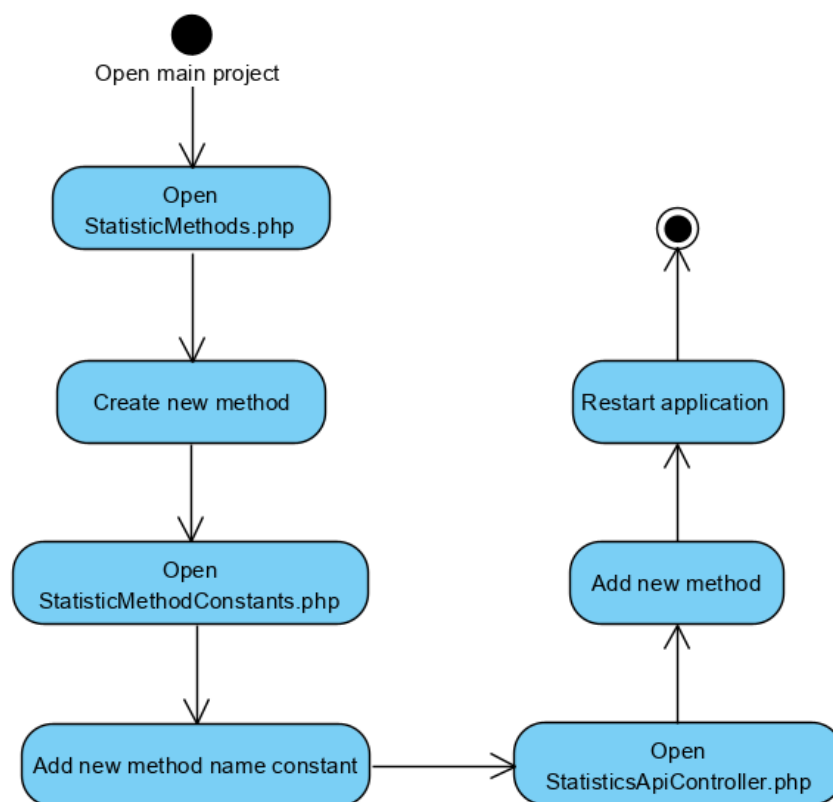


Фиг. 9 Преглед на резултатите от изиграни видео игри за обучение

- Преглед на резултатите от изиграни видео игри за обучение

На Фиг. 9 е представена диаграма на преглед на резултатите от изиграни видео игри за обучение. Всеки потребител на системата може да преглежда резултати на всички играчи, като навигира до меню „Резултати“. Следващата стъпка е да се избере игра и да се натисне бутон „Виж резултати“. Ако потребителят не е избрал игра, системата показва съобщение за грешка. Ако е избрана игра успешно,

потребителят ще бъде навигиран автоматично до таблица с всички резултати на играчите за дадената игра. Потребителят може да разгледа и резултатите на изиграна мини-игра към съответната игра, като избере мини-игра и натисне бутон „Виж резултати“. Отново системата ще покаже съобщение за грешка, ако потребителят не е избрал мини-игра, но е натиснал бутон „Виж резултати“. Ако всичко потребителят е избрал мини-игра, системата автоматично ще отведе потребителя до нова страница с резултатите за мини-играта, която е избрал от предишния екран.

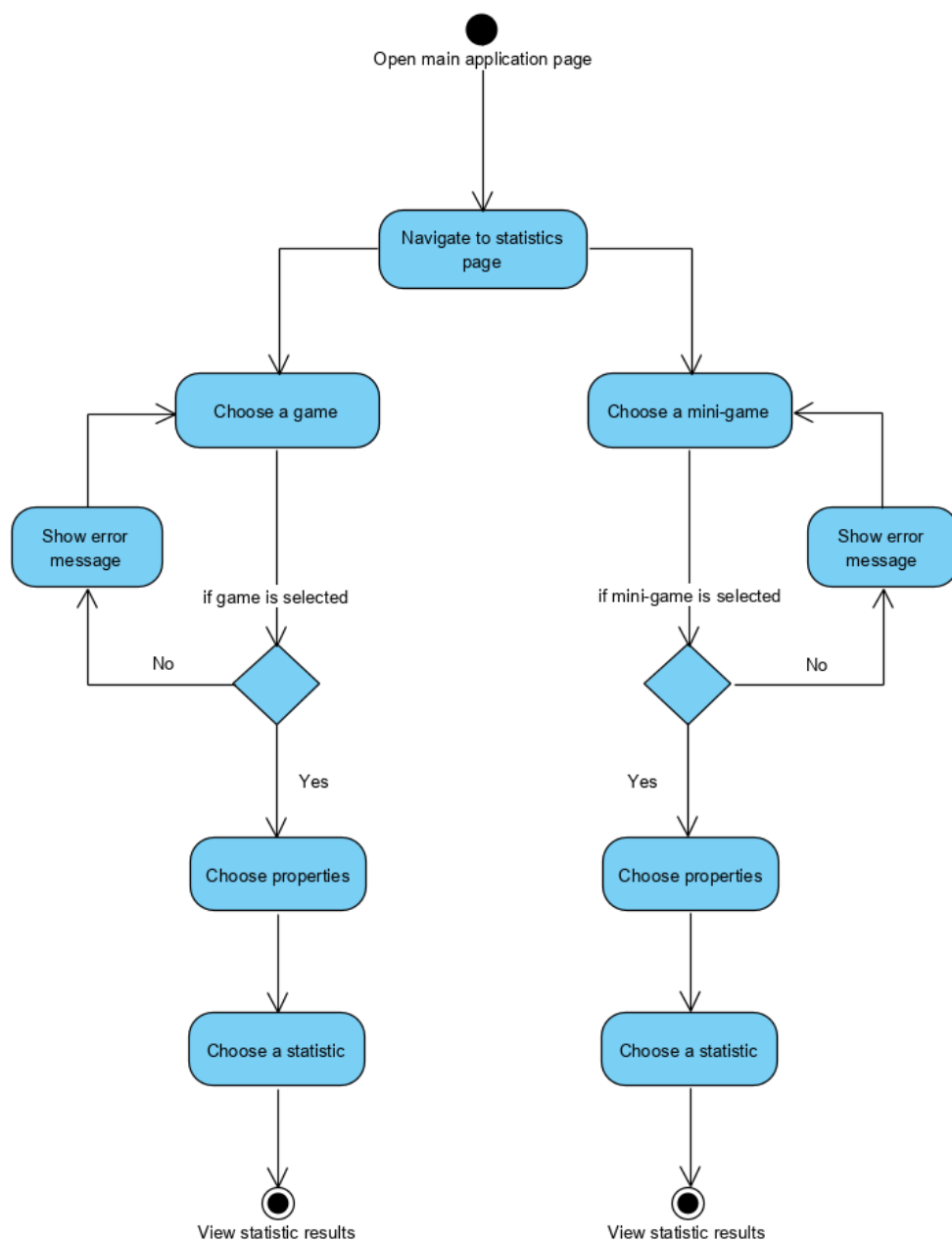


Фиг. 10 Добавяне на нов метод за статистически анализ

- Добавяне на нов метод за статистически анализ

На Фиг. 10 е представена диаграма на добавяне на нов метод за статистически анализ при преглеждането на статистическите резултати. Единствено потребители с роля *администратор* имат правото да добавят нов метод за статистически анализ. Потребителят трябва да отвори файл с име „StatisticMethods.php“, той се намира в директория „app/Services“. В този файл се добавя логиката за имплементиране на статистиката. Следва да се добави в контролера от където данните ще се изпратят до потребителския интерфейс. За целта трябва да се отвори файлът „StatisticsApiController.php“, където се добавя новосъздаденият метод. За да могат имената на методите за статистика да са

унифицирани, трябва да се създадат константни имена, които се дефинират във файла „StatisticMethodConstants.php“. Приложението трябва да се рестартира, за да могат новите промени да се приложат.



Фиг. 11 Преглед на статистически резултати

- Преглед на статистически резултати

На Фиг. 11 е представена диаграма на преглеждане на статистически резултати върху данните от играни видео игри за обучение. Само потребители с роля *учител* и *администратор* имат да достъп до тази функционалност на системата. Потребителят трябва да навигира до меню „Статистики“. Следващите стъпки от процеса са да се избере игра, да се изберат полета (данни), върху които ще се

направи статистическият анализ и последната стъпка е да се избере видът на анализа. Потребителят натиска бутон „Калкулирай“. Ако потребителят е изпуснал някое поле, системата ще покаже съобщение в потребителския интерфейс, че полето е задължително. След успешно калкулиране, системата ще представи статистическите данни на екрана заедно с диаграми показващи стандартното отклонение, стандартната грешка, брой записи и средната стойност за всяко поле избрано в стъпките за генериране на резултатите. На същата страница, може да се направи същият процес за мини-игра. Стъпките са еднакви, разликата е само в това, че се избира мини-игра в първата стъпка. Съобщенията за грешка и резултатите ще се визуализират по същият начин.

4.5. Изводи

В тази глава беше направен обстоен анализ на изискванията на уеб-базирана система за анализ и визуализация на резултати от изиграни видео игри за обучение. Бяха определени, кои са основните действащи лица и какви действия могат да извършват спрямо зададените роли. След това бяха разгледани функционалните (потребителски) изисквания, както и нефункционалните изисквания към системата. Чрез UML диаграми на дейностите бяха разгледани част от най-важните бизнес сценарии. За целта беше използван софтуерният продукт Visual Paradigm. След направения анализ, може да се заключи, че бяха описани всички най-важни стъпки за реализацията на системата, по възможно най-качественият и сигурен начин. Следващата глава от проекта, ще ни запознае с стъпките за проектиране на системата.

Глава 5. Проектиране

Проектиране на един софтуерен проект, по възможно най-качественият, надежден и сигурен начин е сложен процес и изисква задълбочени знания. Трябва да бъде установена цялостната архитектура на системата, интерфейсите, комуникацията между отделните модули и структурата на базата данни. Тези етапи от проектирането на системата са важни, за да може да се осигури безпроблемната работа на системата в дългосрочен план.

5.1. Обща архитектура – слоеве, модули, блокове, компоненти

При разработване на качествени софтуерни системи, изборът на архитектура е изключително важна стъпка за осигуряването на бърза разработка, внедряване и поддръжка на продукта. Също така намалява бъдещите разходи и време за добавяне на нови функционалности, поради отделянето на различните модули в отделни слоеве в системата.

Многослойната архитектура и нейната най-разпространена форма – трислойната (3 – Tier Architecture) е архитектура от тип клиент – сървър, в която потребителският интерфейс, бизнес логиката, съхранението на данни и техният достъп се разработват и поддържат като независими модули, а много често и на различни платформи. Това е една от най-разпространените архитектури при разработването на уеб-базирани приложения [\[31\]](#). Трите слоя в тази архитектура са:

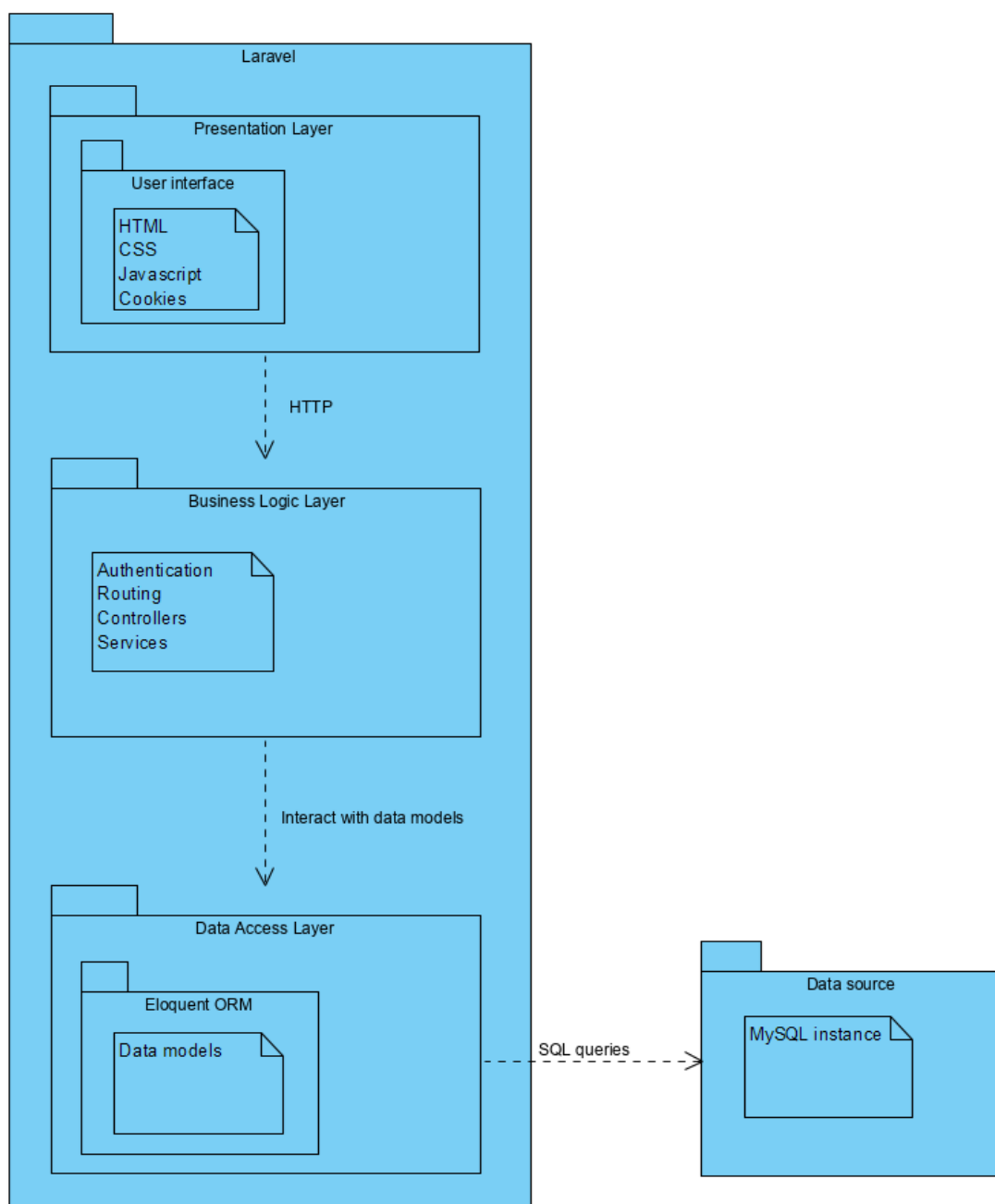
- Презентационен слой
- Слой за бизнес логиката
- Слой за данни

Част от предимствата, които този тип архитектура предлага са:

- Всеки слой от архитектурата е независим от останалите слоеве, като по този начин може да се преработва или добавя нова функционалност без да се засягат другите слоеве. Това спестява време, средства и също така прави интеграцията с други услуги и приложения много по-бърза.
- Сигурността на приложението се подобрява, защото се добавя допълнителен слой между потребителският интерфейс и базата данни.
- Предоставя възможност за споделяне на междинния слой между няколко клиента, като се покачва преизползването и се намалява времето за поддръжка.

- Увеличава се скалируемостта на системата. Всеки слой може да се разработва отделно от останалите и различните модули могат лесно да се заменят с нови, като това води и до по-добра производителност на системата.

В уеб-базираната система за визуализация и анализ на данните от играни видео игри за обучение, се използва технологичната рамка Laravel, която предоставя лесен начин за разделяне на основните слоеве в системата.



Фиг. 12 Софтуерна архитектура на приложението

Избрана е трислойна архитектура. Поради използването на технологична рамка, в проекта са предварително разделени основните слоеве. Всяка част от системата е индивидуална за себе си. Връзката между отделните слоеве се осъществява благодарение на различни протоколи и интерфейси.

Презентационен слой

Това е потребителският интерфейс на системата. Чрез него крайните потребители могат да използват функционалностите на уеб-приложението. За разработването на този слой са използвани HTML, CSS и Javascript. Не е използвана допълнителна технологична рамка, като например Angular, React или Vue, поради факта, че приложението няма сложен интерфейс. Връзката между този слой и бизнес слоя се осъществява чрез HTTP заявки. HTTP заявките от страна на презентационния слой се изпращат и получават благодарение на Javascript библиотеката Axios.

HTTP (Hyper Text Transfer Protocol) представлява протокол за комуникация и пренос на данни през интернет. Той е мрежов протокол от OSI модела. HTTP определя 8 клиентски метода за заявки:

- HEAD
- GET
- POST
- PUT
- DELETE
- TRACE
- OPTIONS
- CONNECT

Най-използваните от тях са GET, POST, PUT и DELETE. Чрез този протокол преносът на данни към и от системата става удобен и структуриран. HTTP протоколът е без състояние, което означава, че на сървърът не се пазят никакви данни от минали заявки. След като потребителят на приложението се е регистрирал и влезнал, всяка заявка от потребителския интерфейс към бизнес логиката, съдържа токен (API token) за автентикация. Този токен има валидност, която се задава от системата и при изтичането му, системата изпраща отговор за грешка за невалидна автентикация. Токенът се пази при клиента под формата на бисквитка (cookie). При заявка от страна на потребителския интерфейс, тези бисквитки могат да бъдат четени от сървъра. Бисквитките могат да предоставят изключително подробна информация за действията на потребителя в уеб-приложението, което ги прави опасни. Важно е да се внимава в какви сайтове разрешаваме използването на бисквитки. [\[32\]](#)

Слой за бизнес логика

В слоя за бизнес логиката се обработват данните в системата. Той служи като междинен слой между потребителският интерфейс и данните съхранявани в базата данни. В него данните се обработват и предават спрямо нуждите на системата. Дефинирани са различни интерфейси за обработване на различни заявки.

Използваният език за програмиране в този случай е PHP заедно с технологичната рамка Laravel. Laravel предоставя възможност за използване на:

- REST - REST или Representational State Transfer е архитектурен стил за осигуряване на стандарти между компютърните системи в мрежата, улеснявайки комуникацията между системите. REST-съвместимите системи, често наричани RESTful системи, се характеризират с това, че са без състояние и отделят грижите на клиента и сървъра. В архитектурния стил REST изпълнението на клиента и изпълнението на сървъра може да се извърши независимо, без единият да знае за другия. Това означава, че кодът от страна на клиента може да бъде променен по всяко време, без това да засяга работата на сървъра, а кодът от страна на сървъра може да бъде променен, без това да повлияе на работата на клиента. REST изисква от своя страна HTTP протокола за пренос на данни. Заявките от клиента към сървъра се обработват от контролери, които имат за цел да извикат нужните класове и методи и да изпратят отговор от заявката обратно. [\[33\]](#)
- Blade шаблони (Template engine) - Шаблонният механизъм свързва модела с данни, обработва кода, даден в шаблоните на източника, и насочва изхода към конкретен текстов файл или поток. Laravel идва с Blade шаблонен механизъм. Това е мощен, лек и предварително инсталиран механизъм за шаблони, който помага на уеб разработчиците да направят процеса на разработка гладък и лесен със своите изключителни оформления. Шаблонният механизъм Blade позволява показване на данни и разширяване на оформления, без това да влияе върху производителността и скоростта на приложението. Той помага да се създават иновативни и невероятни оформления, използвайки функцията за добавяне на съдържание. [\[34\]](#)
- Artisan CLI - Laravel има свой собствен интерфейс за командния ред наречен Artisan (CLI), който позволява да се направи процеса на разработка прост, лесен и бърз. Той използва мощен компонент на Symfony Console за изграждане на приложения. Инструментът също помага при миграции на данни, управление на бази данни и може да се създава скелетен код, модели, контролери и други. [\[35\]](#)

- Eloquent ORM - ORM означава съотнесено към обекта релационно картографиране. Както подсказва името, той позволява да се поддържа лесно взаимодействие с обектите на базата данни с приложения, като се използва красноречив или изразителен синтаксис. Това е един от най - добрите ORM инструменти за уеб разработчици, тъй като им позволява да изпълняват заявки към база данни с прост PHP синтаксис. Така че няма нужда да се отделя много време за писане на сложен код в SQL, което спестява много време. [\[36\]](#)
- Филтри за автентикация – Тези филтри се извикват при постъпване на заявки към сървъра. Тяхната задача е да осигурят ниво на защита, преди заявката да достигне до контролерите на системата, като се направи проверка за автентикация на клиента и оторизация за искания от него ресурс.

Слой за данни

Този слой има грижата за предаването на данни от и към базата данни. Той се състои от сървър с база данни и система за управление на базата данни. Този слой осигурява сигурност и цялост на данните в системата. В технологичната рамка Laravel се използва Eloquent ORM, която прави заявките към базата данни лесна, без да е нужно да се пише SQL код.

Използва се MySQL релационна база данни. Връзката между нея и междинният слой се уществява благодарение на Eloquent ORM. Eloquent управлява свързването с базата данни, изпълнението на заявки и команди. Всяка таблица на база данни има съответния "модел", който се използва за взаимодействие с тази таблица. След като моделът е създаден, може да се извикват различни методи върху него, като част от най-използваните методи за CRUD (Create Read Update Delete) операциите са:

- Създаване на нов запис (Create)

```
$user = new User;

$user->name = 'John';

$user->save();
```

Фиг. 13 Извадка от код за създаване на нов запис в базата данни

- Четене на запис (Read)

```
$users = User::all();
```

```
$users = User::where('votes', '>', 100)->take(10)->get();
```

Фиг. 14 Извадка от код за четене на данни от базата данни

- Промяна на запис (Update)

```
$user = User::find(1);
```

```
$user->email = 'john@foo.com';
```

```
$user->save();
```

Фиг. 15 Извадка от код за промяна на данни от базата данни

- Изтриване на запис (Delete)

```
$user = User::find(1);
```

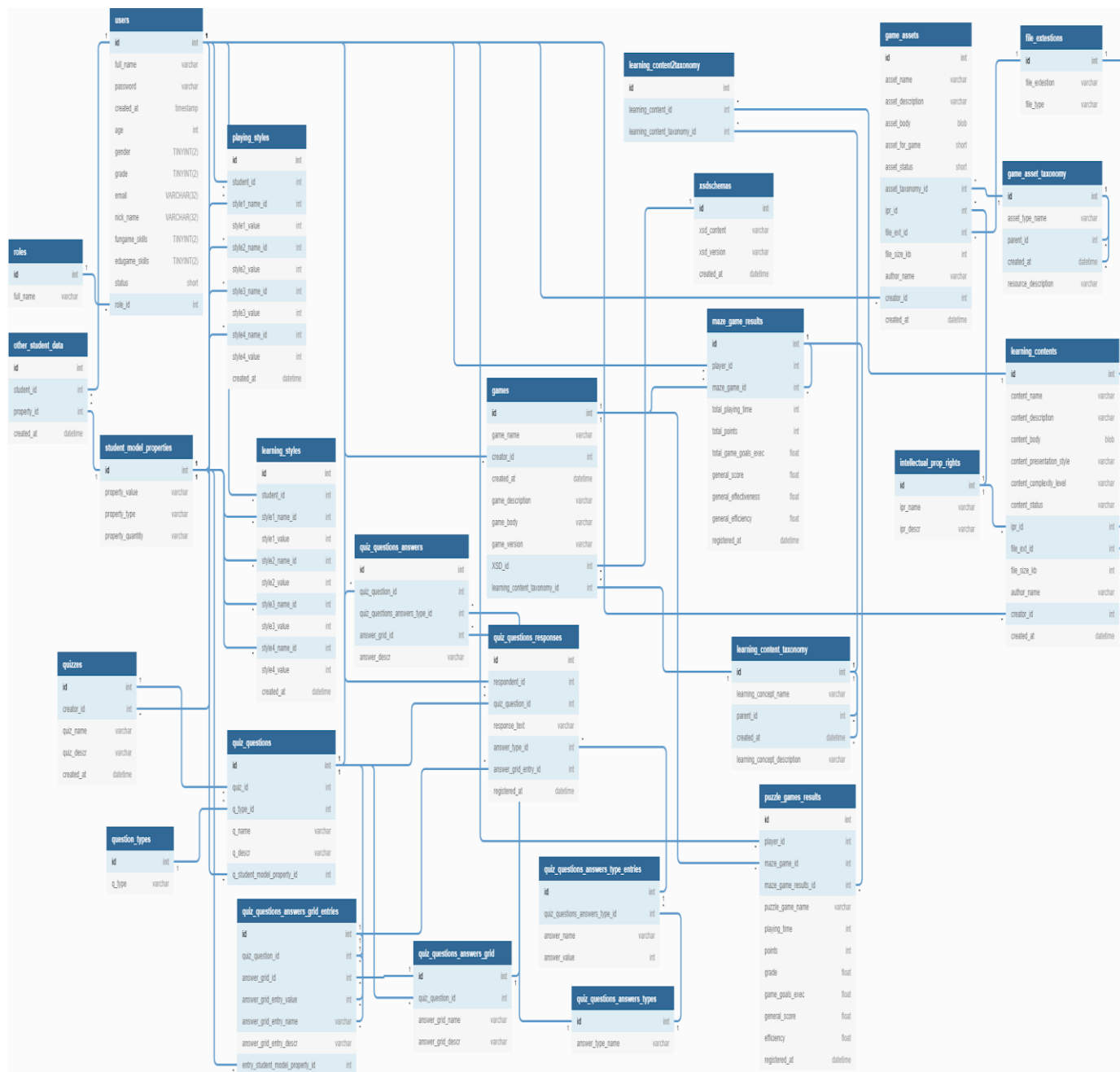
```
$user->delete();
```

Фиг. 16 Извадка от код за изтриване на данни от базата данни

5.2. Модел на данните

Моделът на данните представя обектите, които ще се записват в базата данни. Показват се връзките, техните ограничения и характеристики. Той фундаментално определя по какъв начин данните могат да се съхраняват, организират и манипулират. Използваната в системата база данни MySQL е релационна база данни. Релационната база данни е съвкупност от елементи с данни с предварително определени взаимоотношения между тях. Тези елементи са организирани като набор от таблици с колони и редове. Таблиците се използват за съхраняване на информация за обектите, които да бъдат представени в базата данни. Всяка колона в таблица съдържа определен вид данни и поле, което съхранява действителната стойност. Редовете в таблицата представляват колекция от свързани стойности на един обект. Всеки ред в таблица може да бъде маркиран с уникален идентификатор, наречен първичен ключ (primary key), а редовете между множество таблици могат да бъдат свързани с помощта на

външен ключове (foreign keys). Тези данни могат да бъдат достъпни по много различни начини, без да се реорганизируют самите таблици на базата данни.



Фиг. 17 Основен модел на данните

Диаграма на отношенията на обекти (ERD) показва връзките на набори от обекти, съхранявани в база данни. Обект в този контекст е обект, компонент на данни. Набор от обекти е съвкупност от подобни обекти. Тези обекти могат да имат атрибути, които определят неговите свойства. Тази диаграма показва и взаимовръзките между обектите в базата данни.

За реализацията на уеб приложението за визуализация и анализ на резултати от играни видео игри за обучение са представени основните обекти в базата данни:

- Users (потребител) – Тази таблица описва обекта потребител. Пазят част от най-важните данни за потребителя. Нейните колони са – поле за идентификация (първичен ключ), име, парола, дата на създаване, години, пол, успех, имейл адрес, прякор, умения за учене и играене, статус и идентификатор за роля (външен ключ към таблицата с роли).
- Roles (роли) – Таблица съдържаща всички роли в системата. Описва обекта роля.
- Games (игри) – Таблицата съдържа всички данни за съществуващи игри. Описва обекта игра. Нейните колони са - поле за идентификатор (първичен ключ), име на игра, външен ключ показващ създателя на играта от таблица users, дата на създаване, описание на играта, данни към играта, версия, схема, стил на учене.
- Maze_game_results (резултати от игри лабиринт) – Таблицата съдържа данни за резултатите на играчите от играни игри лабиринт. Описва обекта резултат от игра лабиринт. Колоните, които описват записите са – идентификатор (първичен ключ), външен ключ към идентификатор за играч, външен ключ за идентификатор за игра, общо време, общо точки, процент успешно преминати изпитания, общ резултат, обща ефективност, обща ефикасност, дата на регистриране.
- Puzzle_game_results (резултати от мини-игри от тип пъзел, вградени в залите на лабиринта) – Таблицата съдържа данни за резултатите на играчите от играни игри пъзел. Описва обекта резултат от игра пъзел. Колоните, които описват записите са – идентификатор (първичен ключ), външен ключ към идентификатор за играч, външен ключ за идентификатор за игра, външен ключ за идентификатор за резултат от игра лабиринт, име на игра от тип пъзел, общо време, точки, успех, процент успешно преминати изпитания, общ резултат, ефикасност, дата на регистриране.

Част от обектите в базата данни не предоставят нужна информация за работата на уеб приложението за визуализация и анализ на резултати от играни видео игри за обучение.

Между обектите в базата данни съществуват връзки. Тук са представени част от връзките между обектите:

- Roles – Users -> връзката между таблиците е едно към много. Една роля може да бъде назначена на много потребители.
- Users – Games -> връзката между таблиците е едно към много. Един потребител може да е създал много игри.
- Games – Maze_game_results -> Връзката между таблиците е едно към много. Една игра може да бъде играна много пъти и да има много генерирани резултати.
- Games – Puzzle_game_results -> Връзката между таблиците е едно към много. Една пъзел игра може да бъде играна отново много пъти.
- Maze_game_results – Puzzle_game_results -> Връзката между таблиците е едно към много.

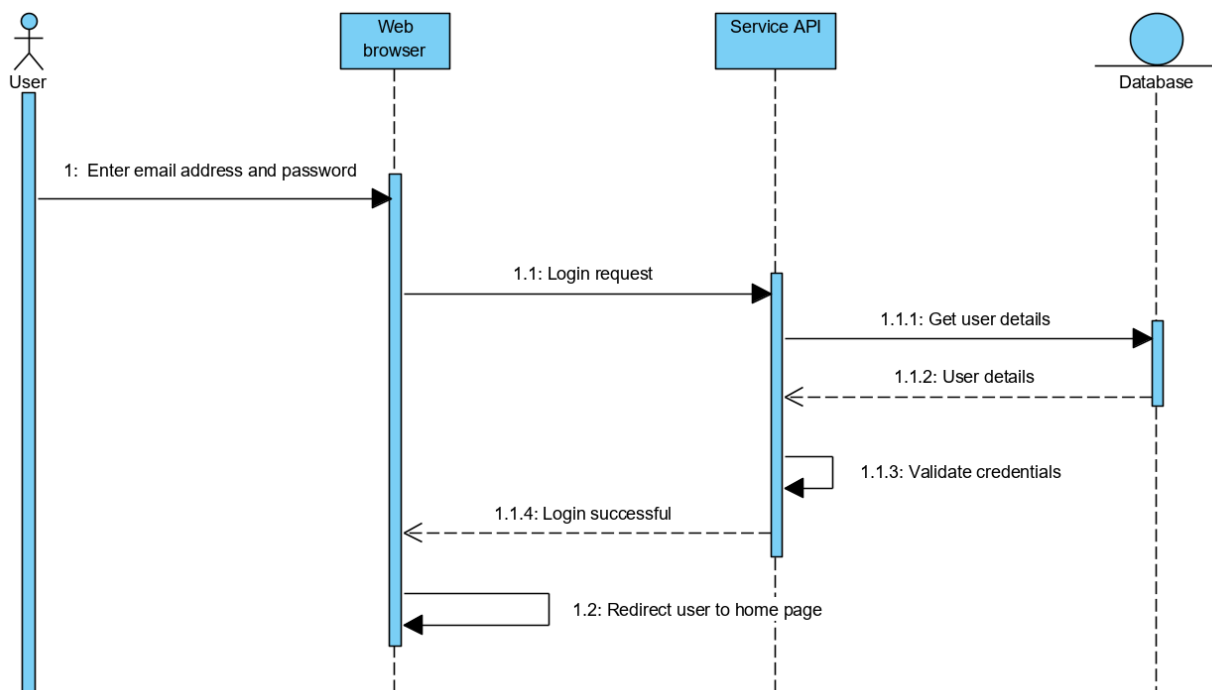
5.3. Диаграми на структура и поведение

UML диаграма е диаграма, базирана на UML (Унифициран език за моделиране) с цел визуално представяне на система заедно с нейните основни участници, роли, действия, артефакти или класове, с цел по-добро разбиране, промяна, поддържане или документиране на информацията за системата. Двете най-широки категории, които обхващат всички останали типове, са поведенческа UML диаграма и Структурна UML диаграма. Както подсказва името, някои UML диаграми се опитват да анализират и изобразят структурата на система или процес, докато други описват поведението на системата, нейните участници и нейните компоненти за изграждане. Диаграмите на поведението показват динамичното поведение на обектите в системата, което може да бъде описано като поредица от промени във времето. Те визуализират, определят, конструират и документират динамичните аспекти на системата. Поведенческите диаграми са категоризирани, както следва: диаграми на случаите на използване, диаграми на взаимодействие, диаграми на състоянието и диаграми на дейностите. [\[37\]](#)

Вход на потребител в системата

На Фиг. 18 е представена диаграма на последователност при вход на потребител в системата и неговата успешна автентикация. При попълване на имейл адрес и парола за достъп, потребителският интерфейс изпраща заявка за вход в системата към приложно-програмен интерфейс (API), който от своя страна валидира получените данни. Сървърът изпраща заявка към базата данни за проверка за съществуващ потребител. При успешно намерен потребител, сървърът получава отговор от базата данни с данните на потребителя и се прави проверка за

автентикация. При успешна автентикация, сървърът изпраща отговор към потребителския интерфейс за успешен вход в системата и пренасочва потребителя към главната страница на веб-приложението.

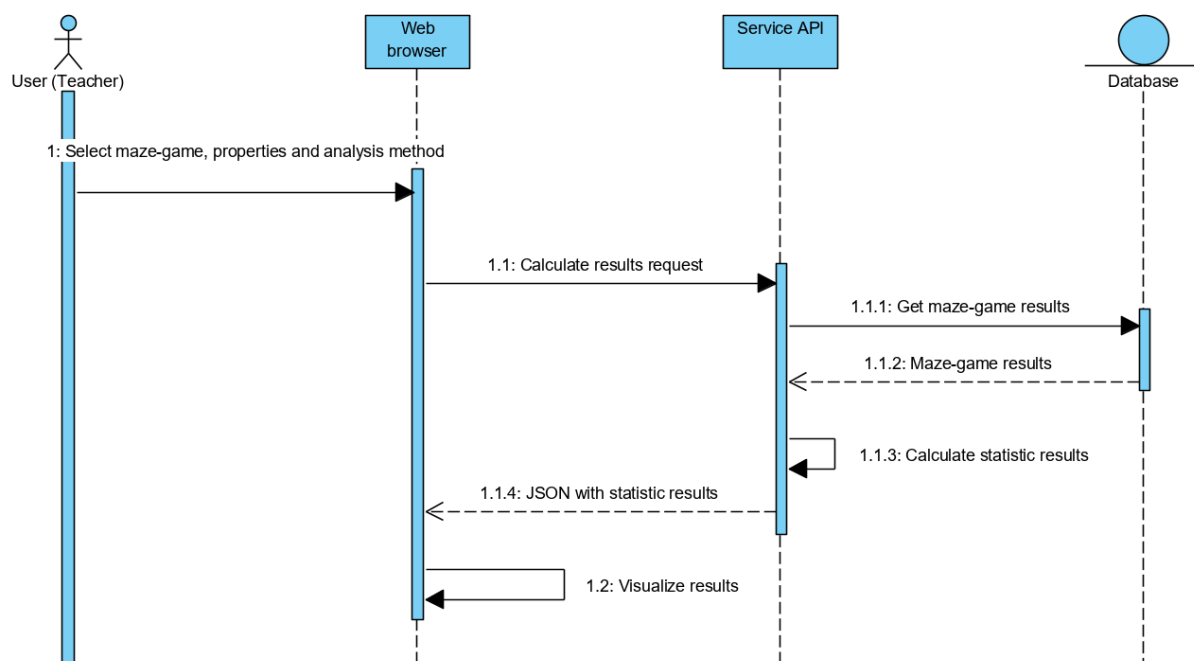


Фиг. 18 Диаграма на последователност при вход на потребител в системата

Преглед на статистически резултати

На Фиг. 19 е представена диаграма на последователност при преглед на статистически резултати от играни видео игри за обучение. Потребителят трябва да се е автентикирал успешно в системата и да има назначена роля *учител*. Следва да се натисне меню „Статистики“ и да се попълнят нужните полета като игра, полета (данни), върху които ще се направи статистическият анализ и вид на анализа. Потребителят натиска бутон „Калкулирай“. Потребителският интерфейс изпраща заявка за калкулиране на резултатите към приложно-програмен интерфейс. Сървърът изпраща заявка към базата данни за получаване на всички резултати от играни пъзел игри (maze-games) от играчите. Данните от базата данни се филтрират според избраните полета от потребителя и се извиква съответният метод за калкулиране на статистическите резултати. След успешно калкулиране, сървърът изпраща данните под формата на JSON към потребителския интерфейс. Статистическите данни се визуализират на екрана заедно с диаграми показващи стандартното отклонение, стандартната грешка, брой записи и средната стойност за всяко поле избрано в стъпките за калкулиране на резултати. Диаграмата за преглед на статистически резултати от играни мини-

игри е същата като тази на Фиг. 19. Важно е да се уточни, че потребители с роля *игрaч*, нямат достъп до тази функционалност на системата.

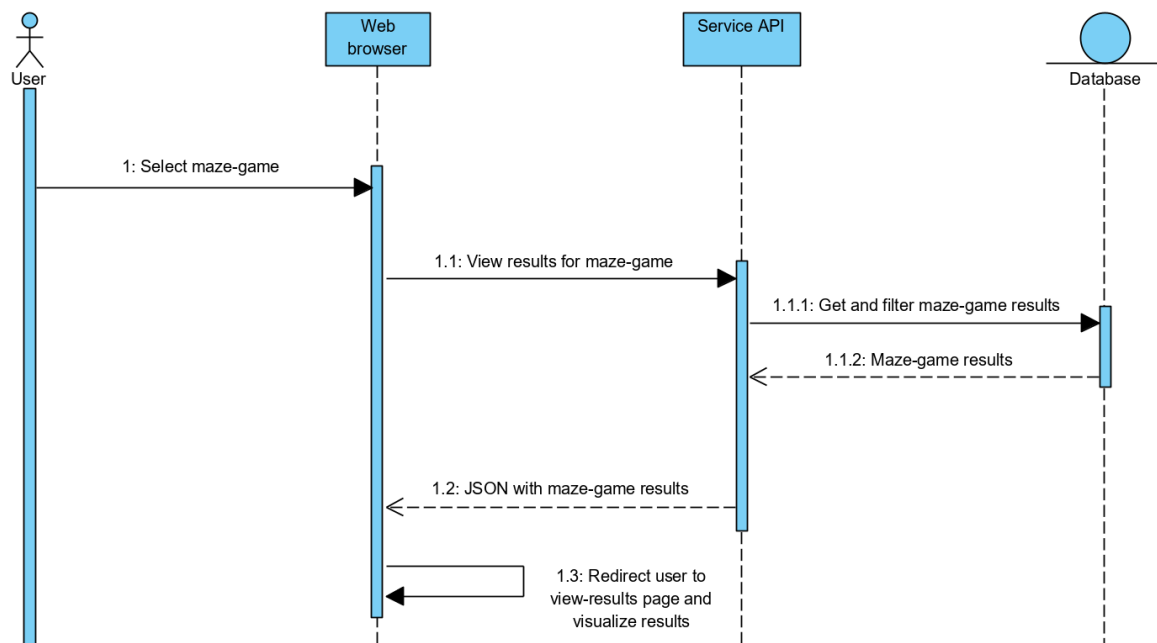


Фиг. 19 Диаграма на последователност при преглед на статистически резултати

Преглед на резултатите от изиграни видео игри за обучение

На Фиг. 20 е представена диаграма на последователност при преглед на резултати от изиграни видео игри за обучение. Потребителят трябва да се е автентикирал успешно в системата. От главното меню трябва да се натисне меню „Резултати“. Потребителят ще бъде навигиран до страницата за избор на игра. След като се избере игра, потребителят натиска бутон „Виж резултати“. Прави се проверка, дали е избрана игра (пъзел-игра). Ако не е избрана игра и е натиснат бутон „Виж резултати“, на екрана се появява съобщение за грешка. При успешно избрана игра потребителският интерфейс изпраща заявка за извличане на резултатите за всеки един играч, изиграл избраната игра към приложно-програмен интерфейс. Сървърът изпраща заявка към базата данни за получаване и филтриране на всички данни за резултати на потребители изиграли дадената игра. След успешно получаване на данните, сървърът изпраща отговор към потребителския интерфейс под формата на JSON и потребителят се автоматично се навигира до „/game-results?gameId=\$“. Където gameId е идентификатор за всяка игра в базата данни. Данните за резултатите се визуализират под формата на таблица с колони – Име, време, точки, успех, ефективност и ефикасност. На същата страница под таблицата с резултати за изиграни видео игри за обучение, могат да се прегледат

и резултати от изиграни мини-игри към дадената игра. Стъпките са идентични с тези за пъзел-игра.

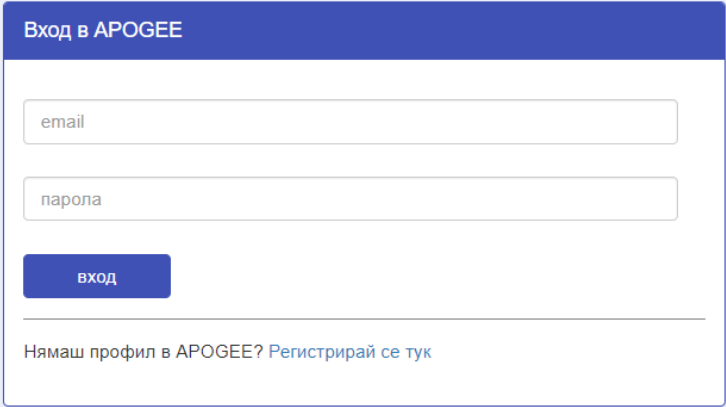


Фиг. 20 Диаграма на последователност при преглед на резултати от изиграни видео игри за обучение

5.4. Потребителски интерфейс

Потребителският интерфейс е една от най-важните части при имплементацията на качествен софтуерен продукт. Той трябва да е лесно използваем, интуитивен и красив. Основната цел е потребителите да достигнат крайната си цел с помощта на приложението възможно най-лесно и бързо.

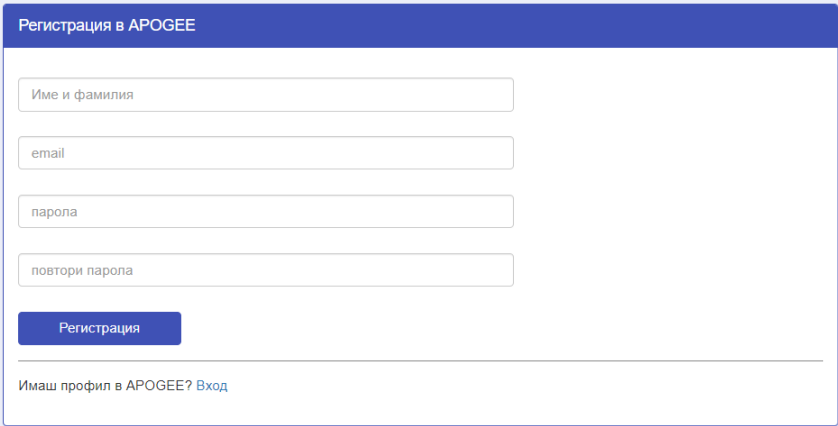
На Фиг. 21 е показана началната страница за вход на потребител в системата.



The image shows a login form titled "Вход в APOGEE". It contains two input fields: "email" and "парола". Below these fields is a blue button labeled "вход". At the bottom of the form, there is a link that says "Нямаш профил в APOGEE? Регистрирай се тук".

Фиг. 21 Страница за вход в системата

На следващата Фиг. 22 е визуализирана страницата за регистрация на потребител в системата. Всички полета са задължителни.



The image shows a registration form titled "Регистрация в APOGEE". It contains four input fields: "Име и фамилия", "email", "парола", and "повтори парола". Below these fields is a blue button labeled "Регистрация". At the bottom of the form, there is a link that says "Имаш профил в APOGEE? Вход".

Фиг. 22 Страница за регистрация на нов потребител в системата

На Фиг. 23 е показата страницата за промяна на данните на профила на потребителя. От тук той може лесно да смени своя имейл адрес и парола за достъп.

Профил в APOGEE

Име и фамилия: Test Test

Email: test@abv.bg

Парола: Парола

Нова парола: Нова парола

Повторете новата парола: Повторете новата парола

Година: 1

Пол: Мъж

Успех: 6

Запази

Фиг. 23 Страница за промяна на данните на профила на даден потребител

На Фиг. 24 е показана страницата за избор на анкета. Показани в списък всички създадени анкети. При избор на някоя от тях, потребителят се навигира до следващата страница за попълване на данните от анкетата.

Анети

Анкета за определяне на профила на потребителя

Анкета за определяне на стил на учене

Анкета за определяне на стил на играене

Анкета за определяне на ниво на потребителско изживяване (UX), game playability & game learnability

Фиг. 24 Страница с всички създадени анкети

На следващата Фиг. 25 е показана страницата с въпросите от избраната анкета от Фиг.24. Тук потребителят попълва данните и натиска бутон „Запази“. Данните се изпращат до сървър, където те биват запазени.

Анкета за определяне на профила на потребителя

Анкетата съдържа няколко въпроса за определяне на профила ти като играч.

1. На колко години си?

2. Какъв е твоят пол?

☐ момче

☐ момиче

3. Какъв е твоят e-mail?

4. Как искаш да се казваш в игрите?

Фиг. 25 Страница с въпроси от избрана анкета

На следващата Фиг. 26 показана страница с избор на игра за визуализиране на резултати от играни видео игри за обучение.

Резултати игри

Total players: 7

Choose a game

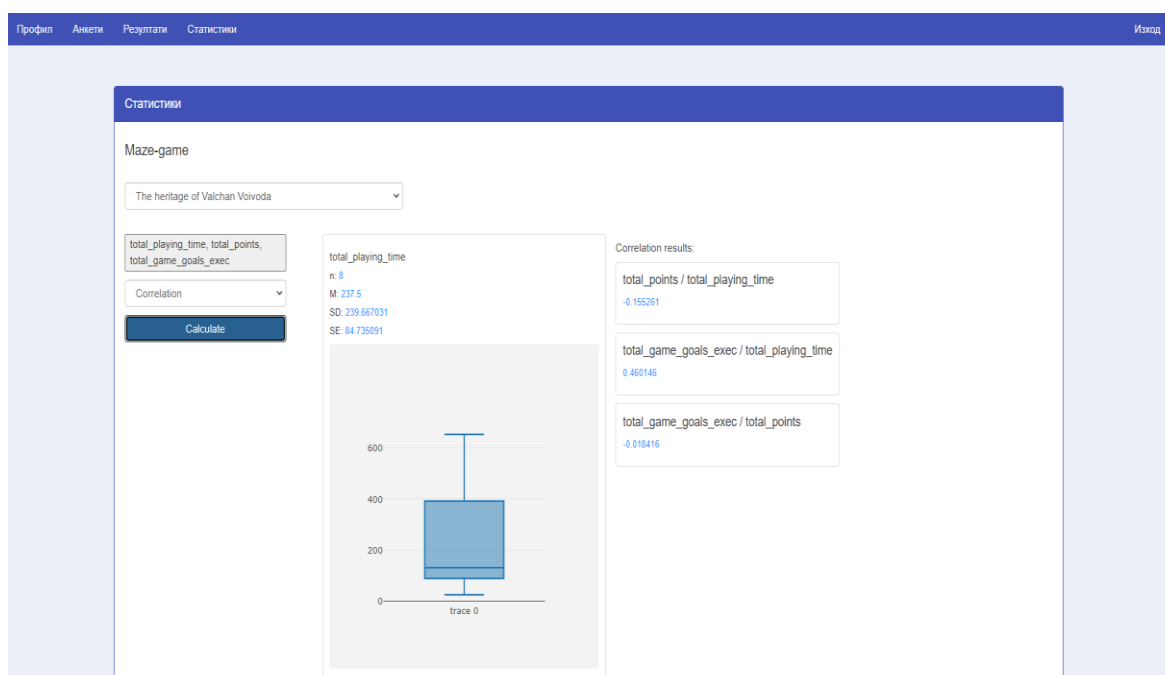
View Results

Фиг. 26 Страница с избор на игра за визуализиране на резултати от играни видео игри за обучение

Следващата Фиг. 27 показва таблица с резултати от играни видео игри за обучение, като таблицата има вградени филтри за персонализирано търсене и сортиране на данните.

Name	Play Time	Points	Score	Goals	Effectiveness	Efficiency
Ivanhoe	137 s	20	14.92 %	10.23 %	73.64 %	84.29 %
Ivanhoe	573 s	225	98.34 %	96.67 %	98.12 %	92.37 %
Ivanhoe	211 s	3211	13.13 %	54.51 %	64.86 %	83.21 %
Ivanhoe	654 s	223	62.14 %	31.46 %	51.48 %	54.34 %
Ivanhoe	24 s	214	13.12 %	32.13 %	12.51 %	32.14 %
Ivanhoe	123 s	3124	12.32 %	23.14 %	72.31 %	32.13 %
Ivanhoe	54 s	21	21.34 %	23.15 %	32.15 %	35.41 %
Ivanhoe	124 s	321	32.45 %	65.43 %	53.14 %	16.54 %

Фиг. 27 Страница за преглед на резултати от играни видео игри за обучение в табличен вид



Фиг. 28 Страница за преглед на статистически резултати от играни видео игри за обучение

На Фиг. 28 е показана страница за преглед на статистически резултати от играни видео игри за обучение. Потребителят трябва да избере нужните полета и да

натисне бутона „Калкулирай“. Системата автоматично ще генерира резултатите под формата на диаграми и колони.

Глава 6. Реализация, тестване/експерименти и внедряване

В тази глава ще се разгледат част от стъпките за завършването на даден софтуерен продукт. Ще се проследи реализирането на модули с извадки от кода, тестването на системата и нейното внедряване. Важно е системата да бъде тествана по време и в края на разработката, за да се осигури стабилност и надеждност.

6.1. Реализация на модулите

В предишната глава беше описана цялостната архитектура на системата, всеки един слой и модулите, които той съдържа, компонентите на базата данни и прилежащите и модели. В тази точка от главата се описва реализацията на всеки модул и извадки от кода, които са част от разработката на цялостната система.

Разгледани са основните два модула на системата, сървърната част и потребителският интерфейс и двата реализирани с помощта на технологичната рамка Laravel. Реализирането на базата данни не е част от дипломният проект, затова няма да бъде разгледана в детайлност, но е важна част от реализирането на цялостната система. Използван е програмният продукт HeidiSQL, с който се осъществява връзката със сървър на база данни на MySQL. С него може лесно и удобно, чрез програмен интерфейс, да се добавя, редактира информация и да се наблюдават, всички данни, които се съдържат в базата данни по всяко време на работата на системата.

Реализиране на сървърната част

Сървърната част на системата е базирана на PHP езика, като за нейната реализация е използвана технологичната рамка Laravel. Първата стъпка за създаването на проекта е инсталирането на технологичната рамка, което може да се направи от [тук](#). За създаването на проекта е използван Composer. Composer е инструмент за управление на зависимости в PHP. Той позволява да се декларират библиотеките, от които зависи вашият проект, и ще ги управлява (инсталира/актуализира) вместо вас. След като сме инсталирали Composer, със следните команди (Фиг.29) създаваме нов проект и стартираме приложението локално. “example-app” е име дадено по подразбиране, трябва да го заместим с желаното име за нашия проект.

```
composer global require laravel/installer

laravel new example-app

cd example-app

php artisan serve
```

Фиг. 29 Инсталиране на Laravel и създаване на нов проект

Чрез командата „php artisan serve“, стартираме приложението локално. Artisan е името на интерфейса на командния ред, включен в Laravel. Той предоставя редица полезни команди за използване при разработването на нашето приложение. Той се задвижва от мощния компонент на Symfony Console. Избрана е версия за PHP 8. Версията на Laravel е 8. На следващата Фиг. 30 са показани всички зависимости на проекта към други пакети. За връзка между потребителският интерфейс и сървърната част е използван Axios.

```
"devDependencies": {
  "axios": "^0.21",
  "laravel-mix": "^6.0.6",
  "lodash": "^4.17.19",
  "postcss": "^8.1.14"
}
```

Фиг. 30 Зависимости на проекта, описани във файл *package.json*

Данните за свързване на приложението с базата данни са описани в *.env* файл. На Фиг. 31 са показани всички нужни данни за връзка с базата данни. Дадена е конфигурация за връзка с локална база данни без парола за достъп, която е използвана само в процес на разработка.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=apogee
DB_USERNAME=root
DB_PASSWORD=
```

Фиг. 31 Данни за връзка с база данни описани във файл *.env*

```

class Game extends Model
{
    protected $table = 'games';
    public $timestamps = false;

    protected $casts = [
        'creator_id' => 'int',
        'XSD_id' => 'int',
        'learning_content_taxonomy_id' => 'int'
    ];

    protected $fillable = [
        'game_name',
        'creator_id',
        'game_description',
        'game_body',
        'game_version',
        'XSD_id',
        'learning_content_taxonomy_id'
    ];

    public function user()
    {
        return $this->belongsTo(related: User::class, foreignKey: 'creator_id');
    }

    public function xsdschema()
    {
        return $this->belongsTo(related: Xdschema::class, foreignKey: 'XSD_id');
    }

    public function learning_content_taxonomy()
    {
        return $this->belongsTo(related: LearningContentTaxonomy::class);
    }

    public function maze_game_results()
    {
        return $this->hasMany(related: MazeGameResult::class, foreignKey: 'maze_game_id');
    }

    public function puzzle_games_results()
    {
        return $this->hasMany(related: PuzzleGamesResult::class, foreignKey: 'maze_game_id');
    }
}

```

Фиг. 32 Изходен код след генериране на модел Game

За менажиране на връзката с базата данни, Laravel използва Eloquent. Eloquent е object-relational mapper (ORM), който прави приятно взаимодействието с база данни. За да може да се менажира базата данни, първо трябва да се генерират всички нужни модели. Това става изключително лесно, чрез инсталиране на пакета „reliese/laravel“. Пакета се инсталира много бързо с помощта на командата „composer require reliese/laravel“. След като пакета се е инсталирал успешно, се

стартира командата `php artisan code:models`. Тази команда ще сканира цялата база данни и ще създаде необходимите модели за връзка с нея.

На Фиг. 32 е даден изходният код след генериране на модела `Game` в папка `app/models`.

Във файла `api.php`, са описани всички пътища използвани за предаване на данни от потребителският интерфейс към сървъра чрез REST APIs. На Фиг. 33 са дадени всички пътища извикващи прилежащите методи в дадени контролери. Всеки контролер представлява клас с методи. При извикване на HTTP GET заявка, могат да се достъпят пътищата, които отговарят за тази функционалност.

```
Route::middleware( middleware: 'auth:api' )->get( uri: '/user', function (Request $request) {
    return $request->user();
});

// REST APIs, called from frontend

// GET
Route::get( uri: 'games', [GameApiController::class, 'getGames'] );
Route::get( uri: 'miniGames/{gameId}', [MiniGameApiController::class, 'getMiniGames'] );

// POST
Route::post( uri: 'statistics/calculateMazeGameResult', [StatisticsApiController::class, 'calculateMazeGameResult'] );
Route::post( uri: 'statistics/calculateMiniGameResult', [StatisticsApiController::class, 'calculateMiniGameResult'] );
```

Фиг. 33 Съществуващи REST APIs в системата

```
class GameApiController extends Controller
{
    public function getGames()
    {
        $games = Game::all();
        if ($games == null) {
            return response()->json( data: 404 );
        }
        return response()->json($games, status: 200);
    }

    public function getGame($id)
    {
        $game = Game::find($id);
        if ($game == null) {
            return response()->json( data: 404 );
        }
        return response()->json($game, status: 200);
    }
}
```

Фиг. 34 Контролер за извличане на създадени игри от базата данни

При заявка за извличане на всички създадени игри в системата се създава HTTP GET заявка към „www.test.com/api/games“. Сървърът приема заявката и извиква метод “getGames” от контролер “GamesApiController” даден на Фиг.34. Методът прави заявка до базата данни за предоставяне на всички данни чрез модела “Game” и метода „all“. При успешно извлечени данни, сървърът изпраща отговор HTTP 200 OK.

На Фиг. 35 е даден метод “calculateMazeGameResult”. Предоставена е изпратената заявката като аргумент към метода. От нея се извличат данните избрани от потребителя чрез потребителския интерфейс. Правят се заявки до базата данни за предоставяне на нужните данни за калкулиране на статистическите резултати. При възникнала грешка при калкулирането на резултатите се изпраща отговор HTTP 400 Bad Request със съобщение от възникналото изключение. При успешно калкулирани статистически резултати се изпраща отговор HTTP 200 OK.

```
public function calculateMazeGameResult(Request $request): JsonResponse
{
    $mazeGameId = $request->params['selectedMazeGameId'];
    $selectedProperties = $request->params['selectedProperties'];
    $selectedMazeGameMethod = $request->params['selectedMazeGameMethod'];

    $propertiesResults = [];

    for ($i = 0; $i < count($selectedProperties); $i++) {
        $property = $selectedProperties[$i];
        $propertyDataFromDb = MazeGameResult::select($property)->where('column: 'maze_game_id', $mazeGameId)->pluck($property)->all();
        if (count($propertyDataFromDb) > 0) {
            $average = array_sum($propertyDataFromDb) / count($propertyDataFromDb);
            $standardDeviation = StatisticMethods::standardDeviation($propertyDataFromDb);
            $standardError = StatisticMethods::standardError($standardDeviation, count($propertyDataFromDb));

            $propertiesResults[$property] = [
                'average' => round($average, precision: 6),
                'standardDeviation' => round($standardDeviation, precision: 6),
                'standardError' => round($standardError, precision: 6),
                'data' => $propertyDataFromDb,
            ];
        }
    }

    $combinations = $this->getAllCombinations($selectedProperties);
    try {
        $mazeGameMethodResults = $this->getMethodResults($selectedMazeGameMethod, $propertiesResults, $combinations);
    } catch (\Exception $e) {
        return response()->json($e->getMessage(), status: 400);
    }

    $response = [
        'propertiesResults' => $propertiesResults,
        'mazeGameMethodResults' => $mazeGameMethodResults,
    ];

    return response()->json($response, status: 200);
}
```

Фиг. 35 Метод за калкулиране на статистически резултати от резултати от играни игри

Методът от Фиг. 35 използва метода *getMethodResults* дадена на Фиг.36. Този метод приема всички данни извлечени преди това от базата данни и манипулира данните с избория от потребителя статистически метод. В класа е реализирана функция за намиране на всички възможни комбинации между избраните параметри. Тези комбинации също биват предадени към метода от Фиг. 35, като всеки статистически метод функционира с точно определен брой параметри. Те биват кодирани константно за всеки отделен статистически метод. Системата има защита от подаване на несъществуващ статистически метод. Ако такъв бъде подаден, се създава (хвърля) изключение и се изпраща отговор до потребителя със съобщение, че исканият статистически метод не е открит.

```
private function getMethodResults($method, $propertiesResults, $combinations): array
{
    $methodResults = [];
    switch ($method) {
        case StatisticMethodsConstants::CORRELATION:
            foreach ($combinations as $combination) {
                if (count($combination) == 2) {
                    $data1 = $propertiesResults[$combination[0]]['data'];
                    $data2 = $propertiesResults[$combination[1]]['data'];
                    $methodResults[join(" / ", $combination)] = round(StatisticMethods::correlation($data1, $data2), 6);
                }
            }
            break;
        case StatisticMethodsConstants::T_TEST:
            foreach ($combinations as $combination) {
                if (count($combination) == 2) {
                    $data1 = $propertiesResults[$combination[0]]['data'];
                    $data2 = $propertiesResults[$combination[1]]['data'];
                    $methodResults[join(" / ", $combination)] = StatisticMethods::tTest($data1, $data2);
                }
            }
            break;
        case StatisticMethodsConstants::ANOVA:
            foreach ($combinations as $combination) {
                if (count($combination) == 3) {
                    $data1 = $propertiesResults[$combination[0]]['data'];
                    $data2 = $propertiesResults[$combination[1]]['data'];
                    $data3 = $propertiesResults[$combination[2]]['data'];
                    $methodResults[join(" / ", $combination)] = StatisticMethods::anovaOneWay($data1, $data2, $data3);
                }
            }
            break;
        case StatisticMethodsConstants::EFFECT_SIZE:
            foreach ($combinations as $combination) {
                if (count($combination) == 2) {
                    $data1 = $propertiesResults[$combination[0]]['data'];
                    $data2 = $propertiesResults[$combination[1]]['data'];
                    $methodResults[join(" / ", $combination)] = round(StatisticMethods::effectSizeCohensD($data1, $data2), 6);
                }
            }
            break;
        default:
            throw new \Exception('Statistic method not found');
    }
    return $methodResults;
}
```

Фиг. 36 Метод за калкулиране на статистическите резултати за избран статистически метод чрез манипулиране на извлечените данни от базата данни

На следващата Фиг. 37 е представен класът *StatisticMethods*. Този клас съдържа всички съществуващи методи в системата. Промяната и добавянето на нов вид статистически метод става много бързо и удобно, защото цялата логика за

манипулиране на данните е централизирана в този клас и прилежащите му методи. Системата използва за пресмятане на статистическите резултати библиотеката MathPHP. Тази библиотека има набор от най-различни методи за статистика.

```
class StatisticMethods
{
    public static function standardDeviation($data1): float
    {
        return Descriptive::standardDeviation($data1);
    }

    public static function standardError($standardDeviationValue, $countSquared): float
    {
        return $standardDeviationValue / sqrt($countSquared);
    }

    public static function correlation($data1, $data2): float
    {
        return Correlation::sampleCorrelationCoefficient($data1, $data2);
    }

    public static function tTest($data1, $data2): string
    {
        $result = Significance::tTestTwoSample($data1, $data2);
        return 'P1: ' . round($result['p1'], precision: 6) .
            '<br>' .
            'P2: ' . round($result['p2'], precision: 6) .
            '<br>' .
            'T Score: ' . round($result['t'], precision: 6) .
            '<br>' .
            'DF: ' . round($result['df'], precision: 6);
    }

    public static function anovaOneWay($data1, $data2, $data3): string
    {
        $result = ANOVA::oneWay($data1, $data2, $data3)['ANOVA']['treatment'];
        return 'F: ' . round($result['F'], precision: 6) .
            '<br>' .
            'P: ' . round($result['P'], precision: 6);
    }

    public static function effectSizeCohensD($data1, $data2): float
    {
        $data1M = array_sum($data1) / count($data1);
        $data2M = array_sum($data2) / count($data2);
        $data1SD = self::standardDeviation($data1);
        $data2SD = self::standardDeviation($data2);
        return EffectSize::cohensD($data1M, $data2M, $data1SD, $data2SD);
    }
}
```

Фиг. 37 Клас *StatisticMethod* съдържащ всички съществуващи статистически методи в системата

Реализиране на потребителския интерфейс

Потребителският интерфейс на приложението е реализиран с помощта на Laravel и по точно Blade. Blade е прост, но мощен шаблон, включен в Laravel. За разлика от някои PHP шаблони, Blade не ви ограничава да използвате обикновен PHP код в шаблоните. Всъщност всички шаблони на Blade се компилират в обикновен PHP код и се кешират, докато не бъдат променени, което означава, че Blade не добавя никакво забавяне към приложението. Файловете с шаблони на Blade използват разширението на файла `.blade.php` и се съхраняват в директорията `resources/views`. Blade страниците могат да се изпращат от маршрути или контролер с помощта на глобалния метод `view`. Първоначалните данни за всяка страница могат да бъдат прехвърлени с помощта на втория аргумент на `view` метода.

Всяка страница в проекта има прилежащ контролер и метод, който изпраща обратно към потребителя исканата страница с първоначалните данни към нея. Laravel използва свой собствен алгоритъм за намиране на исканата страница, като се подава името на файла преди разширението `.blade.php`.

На Фиг. 38 е показан метод от контролер *StatisticsController*, който наследява класа *BaseController*. Методът връща като отговор исканата страница с първоначалните данни прилежащи към нея.

```
public function statisticsPage()
{
    $games = Game::all();
    $mazeGameResultsColumnNames = array_keys(MazeGameResult::first()->getAttributes());
    $mazeGameColumnNames = array_filter($mazeGameResultsColumnNames, function ($columnName) {
        return (strpos($columnName, 'id') === false && strpos($columnName, 'registered') === false);
    });
    $puzzleGameResultsColumnNames = array_keys(PuzzleGameResult::first()->getAttributes());
    $miniGameColumnNames = array_filter($puzzleGameResultsColumnNames, function ($columnName) {
        return (strpos($columnName, 'id') === false && strpos($columnName, 'registered') === false && strpos($columnName, 'registered') === false);
    });

    $methods = StatisticMethodsConstants::METHODS_ARRAY;

    return view('statistics.statistics', [
        'mazeGames' => $games,
        'mazeGameColumnNames' => $mazeGameColumnNames,
        'miniGameColumnNames' => $miniGameColumnNames,
        'methods' => $methods
    ]);
}
```

Фиг. 38 Метод *statisticPage()* предоставящ страницата Statistics с прилежащите първоначални данни

Освен PHP, за реализирането на дадена страница е нужно използването и на HTML и CSS. HTML структурира изгледа на страницата или накратко казано, скелета на страницата. На Фиг. 39 е дадена част от страницата *statistics.blade.php* предназначена за визуализиране на статистическите резултати. Чрез къдрави скоби и знака за долар „\$“ могат да се достъпват данните подадени от сървърната

част към потребителския интерфейс. Използвани за HTML елементи за избор на елемент от падащо лист (таг `<select>`), както и елемент за създаване на бутон (таг `<button>`). Стилизацията е извършена, чрез подаване на CSS класове към дадените тагове и е разделена от HTML кода. Връзката между отделните елементи в *.blade.php* файловете и JavaScript функциите е реализиране чрез достъпване на елементите чрез идентификатори (id).

```
<div style="...">
  <select id="selectMazeGames" class="form-control" style="..."
    onchange="onSelectMazeGame()">
    <option value="" selected>Select a maze-game</option>
    @foreach ($mazeGames as $mazeGame)
      <option value="{{ $mazeGame->id }}">{{ $mazeGame->game_name }}</option>
    @endforeach
  </select>

  <div id="mazeGameError" class="alert alert-danger" style="...">
    <ul>
      <li>Please select a maze-game</li>
    </ul>
  </div>
</div>

<div>
  <div class="inline-block content-to-top">
    <div class="custom-margin-bottom" style="...">
      <select id="mazeGameMultiselect" multiple="multiple">
        @foreach ($mazeGameColumnName as $mazeGameColumnName)
          <option value="{{ $mazeGameColumnName }}">{{ $mazeGameColumnName }}</option>
        @endforeach
      </select>
    </div>

    <div class="custom-margin-bottom" style="...">
      <select id="mazeGameMethod" class="margin-right form-control">
        <option value="" selected>Select a method</option>
        @foreach ($methods as $method)
          <option value="{{ array_search($method, $methods) }}">{{ $method }}</option>
        @endforeach
      </select>
    </div>

    <div>
      <button class="btn btn-primary" style="..." type="button" onclick="calculateMazeGame()">
        Calculate
      </button>
    </div>
  </div>
</div>
```

Фиг. 39 Част от страница *statistics.blade.php*

Всяка страница има прилежащ JavaScript файл с функции, като част от които извършват заявки към базата данни. За да се постигне реактивност в приложението и да не се презарежда цялата страница за всяка заявка към сървъра

се използва библиотеката Axios. Това е HTTP клиент, който предоставя функционалност за комуникация със сървър чрез обещания (Promises). На Фиг. 40 е представена функция от файл *statistic.js* извършваща HTTP POST заявка към сървъра за калкулиране на статистическите резултати. Като параметри към заявката се подава идентификатора на избраната игра (*selectedMazeGameId*), избраният статистически метод (*selectedMazeGameMethod*) и избраните полета от базата данни (*selectedProperties*). При грешка функцията изчиства резултатите от потребителския интерфейс.

```
async function getMazeGameResult(selectedMazeGameId, selectedMazeGameMethod, selectedProperties) {
  try {
    const response = await axios.post('api/statistics/calculateMazeGameResult', {
      data: {
        params: {
          selectedMazeGameId: selectedMazeGameId,
          selectedMazeGameMethod: selectedMazeGameMethod,
          selectedProperties: selectedProperties,
        }
      }
    });
    console.log(response, 'response maze game result');
    return response.data;
  } catch (error) {
    console.error(error);
    let propertyResultsDiv = $('#containerMazePropertiesResults');
    while (propertyResultsDiv[0].firstChild) {
      propertyResultsDiv[0].removeChild(propertyResultsDiv[0].firstChild);
    }
    let methodResultsDiv = $('#mazeGameMethodResults');
    while (methodResultsDiv[0].firstChild) {
      methodResultsDiv[0].removeChild(methodResultsDiv[0].firstChild);
    }
  }
}
```

Фиг. 40 Функция *getMazeGameResult* от файл *statistic.js*

6.2. Планиране на тестването

Тестването на системата е важно да премине през възможно най-много сценарии за валидиране на данните от страна на сървъра и това, което се получава в клиентската част. Важно е да се отбележат и по рядко срещани сценарии, за да могат да се избегнат бъдещи грешки в системата. Системата трябва да известява в случай на възникнала грешка. Тестването служи за подsigуряване на стабилността на системата и нейната устойчивост.

6.3. Модулно и системно тестване

Тестването се изпълнява чрез ръчно изпълнение на различни видове сценарии, които потребителите биха извършили при работа в системата.

Таблица 4. Тестови сценарии

Очакван резултат	Стъпки за изпълнение
Успешно регистриране в системата	<ol style="list-style-type: none"> 1. Зареждане на началната страница 2. Натискане на бутон „Регистрирай се тук“ 3. Въвеждане на име и фамилия, имейл, парола и повтаряне на паролата 4. Натискане на бутон „Регистрация“
Неуспешно регистриране в системата	<ol style="list-style-type: none"> 1. Зареждане на началната страница 2. Натискане на бутон „Регистрирай се тук“ 3. Въвеждане само на име и фамилия без имейл 4. Натискане на бутон „Регистрация“ 5. Извеждане на съобщение за грешка
Успешен вход в системата	<ol style="list-style-type: none"> 1. Зареждане на началната страница 2. Въвеждане на имейл и парола във формата 3. Натискане на бутон „Вход“
Неуспешен вход в системата	<ol style="list-style-type: none"> 1. Зареждане на началната страница 2. Въвеждане на имейл и грешна парола във формата 3. Натискане на бутон „Вход“ 4. Извеждане на съобщение за грешка
Преглед на резултати от играни видео игри	<ol style="list-style-type: none"> 1. Избор на меню „Резултати“ от лентата с менюта 2. Избора на игра от падащия лист 3. Натискане на бутон „Виж резултати“ 4. Зареждане на таблица с всички резултати на играчи играли избраната игра
Съобщение за грешка при преглед на резултати от играни видео игри без избор на игра	<ol style="list-style-type: none"> 1. Избор на меню „Резултати“ от лентата с менюта 2. Натискане на бутон „Виж резултати“ 3. Извеждане на съобщение за грешка
Преглед на резултати от играни мини-игри	<ol style="list-style-type: none"> 1. Избор на меню „Резултати“ от лентата с менюта 2. Избор на игра от падащия лист 3. Натискане на бутон „Виж резултати“

	<ol style="list-style-type: none"> Зареждане на страница с таблица и под нея форма за избиране на прилежаща мини-игра Избор на мини-игра от падащия лист Натискане на бутон „Виж резултати“ Зареждане на нова таблица с всички резултати на играчи играли избраната мини-игра
Съобщение за грешка при преглед на резултати от играни мини-игри без избор на игра	<ol style="list-style-type: none"> Избор на меню „Резултати“ от лентата с менюта Избор на игра от падащия лист Натискане на бутон „Виж резултати“ Зареждане на страница с таблица и под нея форма за избиране на прилежаща мини-игра Натискане на бутон „Виж резултати“ Извеждане на съобщение за грешка
Преглед на статистически резултати	<ol style="list-style-type: none"> Проверка на потребителя за роля <i>учител</i> Избор на меню „Статистики“ от лентата с менюта Избор на игра от падащия лист Избор на полета Избор на метод за анализ на данните Натискане на бутон „Калкулирай“ Зареждане на лист с диаграми и отдясно до тях резултати от анализа
Съобщение за грешка при преглед на статистически резултати при липса на избрана игра, полета или метод за анализ на данните	<ol style="list-style-type: none"> Проверка на потребителя за роля <i>учител</i> Избор на меню „Статистики“ от лентата с менюта Натискане на бутон „Калкулирай“ Извеждане на съобщение за грешка
Излизане от системата	<ol style="list-style-type: none"> Проверка дали потребителят е влязъл в системата Натискане на бутон „Изход“ от лентата с менюта

	3. Потребителят е успешно излязъл от системата
--	--

6.4. Анализ на резултатите от тестването

Всички тестови сценарии показани в Таблица 4. са изпълнени успешно. Всички резултати съвпадат с очакваните резултати от системата при изпълнение на всички описани стъпки. Тестовите сценарии са валидни при нормална употреба на системата и при възможни грешни действия.

6.5. Експериментално внедряване

Създадената софтуерна платформа за анализ и визуализация на резултати от видео игри за обучение беше практически внедрена посредством инсталиране и тестване на публичен Уеб сървър, поддържан от компанията СуперХостинг.БГ в рамките на научно-изследователския проект АПОГЕЙ. Използваният хостинг план „СуперПро“ включва поддръжка на домейн (apogee.online), защитен достъп хостинг сървър, DNS и е-мейл сървъри, SSH достъп и поддръжка на PHP с Laravel и MySQL. В сътрудничество с част от екипа на проект АПОГЕЙ, последователно бяха създадени базата данни, бе настроен и запуснат Laravel, и бяха инсталирани PHP скриптовете със съответните настройки.

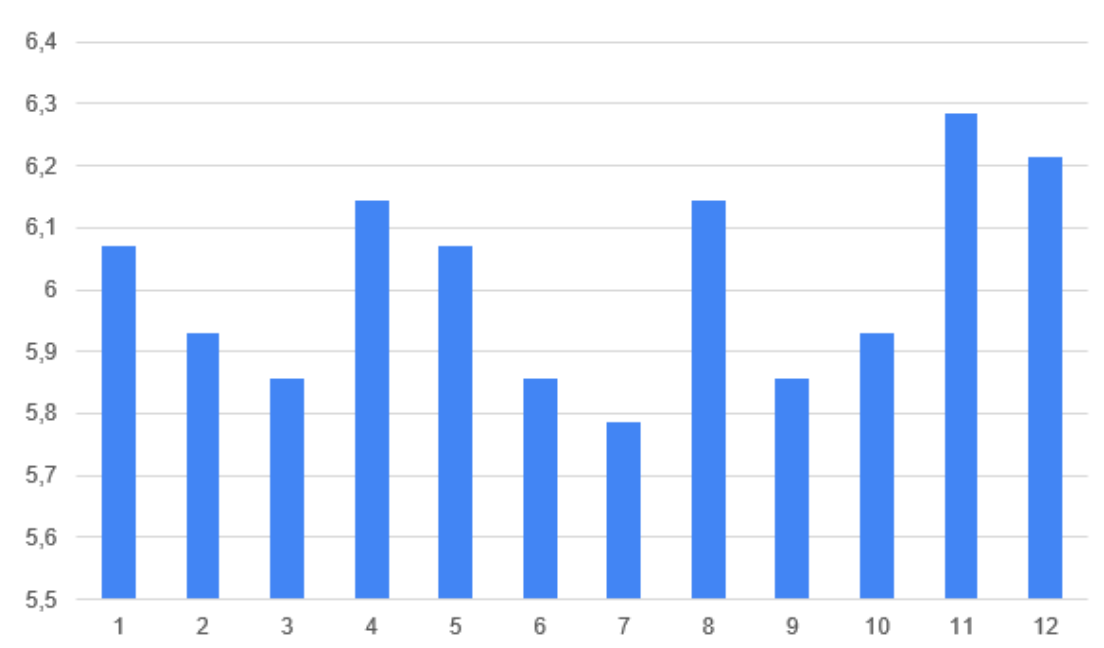
Софтуерната платформа за анализ и визуализация на резултати от видео игри за обучение е налична на адрес http://apogee.online/apogee_new/. При достъп до този адрес, потребителят се пренасочва автоматично към страницата http://apogee.online/apogee_new/login, където може да се извърши оторизиран вход в системата или регистрация. За тестване може да се използва тестов потребител с потребителско име test@test.com и парола test. След влизане се показва страница с четири анкети – за определяне на профила на потребителя, за определяне на стил на учене, за определяне на стил на играене, и за определяне на ниво на потребителско изживяване (UX), game playability & game learnability. Освен анкетите, менюто предлага редакция на профила, преглед на резултати, на статистики и изход.

6.6. Анкета за оценяване на използваемостта

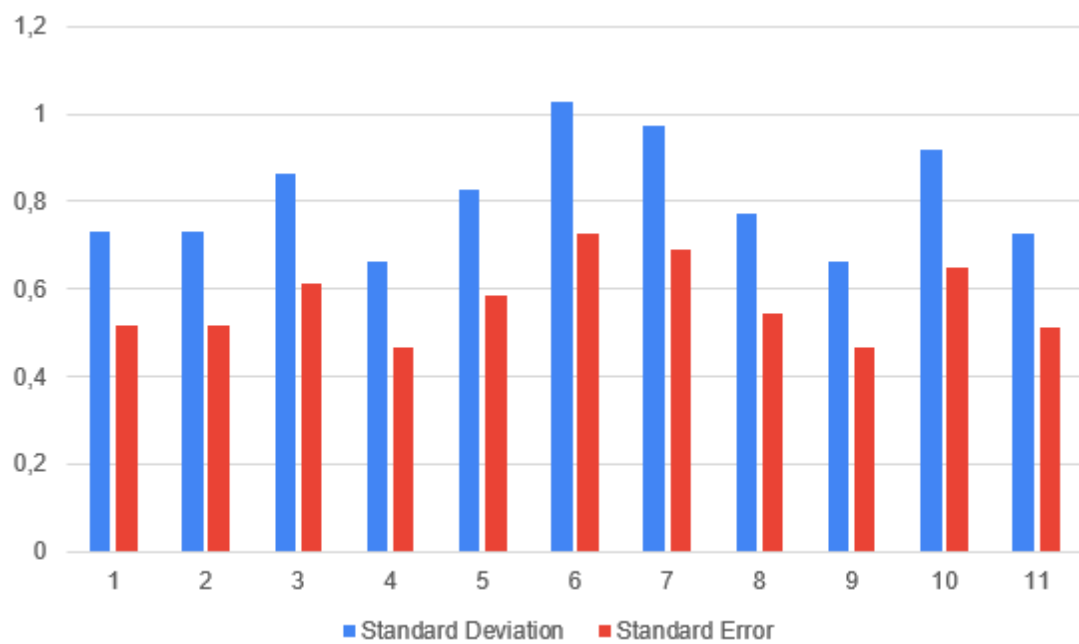
Създадена е анкета, в която има въпроси свързани с оценка на функционалностите на системата и използваемостта ѝ. Достъпът до анкетата е през този линк: https://docs.google.com/forms/d/e/1FAIpQLSfXGjypKQCvsYOVy8cSIUQs6s_G8-BHDzKWO_LjiCODruhW0w/viewform?usp=sf_link. Използвани са основните качествени изисквания за изправна софтуерна система и използване на един от

най-популярните въпросници за изследване на използваемост на софтуерни приложения - Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology, описани в [Davis, F. D. (1989) Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. MIS Quarterly, 13:3, 319-340]. За целта е използвана 7-степенна Ликертова скала като 1 – Определено не, а 7 – Определено да). [38]

Чрез Google Forms е създадена онлайн анкета с 12 въпроса. В началото на анкетата е представено кратко описание на системата и линк към системата качена на: http://apogee.online/apogee_new/. Изпратена е на 16 човека да участват в изследването на използваемостта на системата и като 12 от тях са я попълнили успешно. За всеки въпрос е пресметната средна стойност ($M=Mean$) показана на Фиг. 41, стандартна грешка ($SE=Standard Error$) и стандартно отклонение ($SD=Standard Deviation$) показани на Фиг. 42. Използвани са резултатите от анкетата, с които да се изчислят стойностите. Установени са изводи на база на анализа на дескриптивните характеристики на отговорите на въпросите от анкетата, че използваемостта на създадената платформа за анализ и визуализация на резултати от играни видео игри за обучение е оценена като много висока.



Фиг. 41 Средна стойност, измерена по въпроси



Фиг. 42 Стандартно отклонение и стандартна грешка, измерени по въпроси

Глава 7. Заключение

7.1. Обобщение на изпълнението на началните цели

В началото на разработката на даденото уеб-приложение бяха зададени цели, които съобразено с времето за изпълнение, бяха постигнати напълно.

Беше направен сравнителен анализ на вече съществуващи решения, в които липсва интеграция със самите видео игри за обучение. Изложени са предимствата за изграждането на такава система и какви функционалности тя трябва да предоставя на своите потребители.

Бяха разгледани различни видове софтуерни технологии за реализиране на уеб-базирани приложения. Чрез сравнителен анализ бяха представени техните предимства и недостатъци и на база това бяха избрани най-подходящите за целите на приложението.

Беше направен анализ на потребителските и качествените изисквания и тяхното визуализиране чрез различни видове диаграми. Тук бяха засегнати какви функционалности трябва да може да съдържа системата и различните характеристики или ограничения на имплементацията на софтуера, неговата поддръжка и държание в различни ситуации.

Проектирането на системата е основна част от бъдещото развитие на продукта. Затова беше установена цялостната архитектура на системата, интерфейсите, комуникацията между отделните модули и структурата на базата данни.

Чрез диаграми на последователност, беше представено динамичното поведение на обектите в системата и тяхната поредица от промени във времето. Представен бе потребителският интерфейс и бяха тествани всички най-често срещани сценарии при използването на уеб-приложението.

Практически експеримент беше направен с разработената софтуерна платформа за анализ и визуализация на играни видео игри за обучение, последван от анкетиране на потребителите относно използваемостта на системата. Резултатите от експеримента бяха анализирани и представени чрез графики, показващи стандартното отклонение и стандартната грешка.

В заключение може да се каже, че системата отговаря на всички изисквания и е полезен инструмент при анализиране и визуализация на резултати от играни видео игри за обучение.

7.2. Насоки за бъдещо развитие и усъвършенстване

В първоначалната версия на системата са реализирани промяна на данните на профила на потребителя, попълване на анкети, преглед на резултати от играни видео игри за обучение и преглед и визуализация на статистически резултати.

В следващата версия на системата могат да се добавят следните функционалности, които ще подобрят използваемостта и ще дадат повече информация за усвояването на умения и знания от обучаващите се:

- Добавяне на времеви диапазони за преглед на статистически резултати от играни видео игри за обучение
- Добавяне на повече статистически методи
- Добавяне на снимка към профила на потребителя
- Стартиране на игра от платформата за анализ и визуализация на резултати от играни видео игри за обучение
- Разработване на мобилна версия на платформата
- Запис на предварително избрани полета за анализ на резултатите от играни видео игри за обучение

Използвана литература

- [1] “Software Instruments for Analysis and Visualization of Game-Based Learning Data”, последно достъпен на 17.10.2021г.
- [2] “СЕРИОЗНИТЕ ИГРИ – ИНОВАТИВНО СРЕДСТВО ЗА ОБУЧЕНИЕ”, <http://fmi-plovdiv.org/GetResource?id=2855>, последно достъпен на 17.10.2021г.
- [3] “What are serious games?”, <https://grendelgames.com/what-are-serious-games/>, последно достъпен на 17.10.2021г.
- [4] “Standard Deviation: Definition, Examples”, <https://www.statisticshowto.com/probability-and-statistics/standard-deviation/>, последно достъпен на 17.10.2021г.
- [5] ДА ПРЕОТКРИЕМ СТАТИСТИКАТА
С IBM SPSS STATISTICS, Българска Първо издание, Copyright© 2016,
Zornitza Ganeva
- [6] “ANOVA Test: Definition, Types, Examples, SPSS”, <https://www.statisticshowto.com/probability-and-statistics/hypothesis-testing/anova/>, последно достъпен на 17.10.2021г.
- [7] “Cohen’s D: Definition, Examples, Formulas”, <https://www.statisticshowto.com/cohens-d/>, последно достъпен на 17.10.2021г.
- [8] “Excel Definition”, <https://corporatefinanceinstitute.com/resources/excel/study/excel-definition-overview/>, последно достъпен на 17.10.2021г.
- [9] “MATLAB”, <https://www.mathworks.com/products/matlab.html>, последно достъпен на 17.10.2021г.
- [10] “The R Project for Statistical Computing”, <https://www.r-project.org/>, последно достъпен на 17.10.2021г.
- [11] “IBM SPSS software”, <https://www.ibm.com/analytics/spss-statistics-software>, последно достъпен на 17.10.2021г.
- [12] “Comparison of data analysis packages: R, Matlab, SciPy, Excel, SAS, SPSS, Stata”, <https://brenocon.com/blog/2009/02/comparison-of-data-analysis-packages-r-matlab-scipy-excel-sas-spss-stata/>, последно достъпен на 17.10.2021г.
- [13] “PhpStorm”, <https://www.jetbrains.com/phpstorm/>, последно достъпен на 17.10.2021г.
- [14] “Visual Studio”, <https://visualstudio.microsoft.com/>, последно достъпен на 17.10.2021г.

- [15] “Eclipse”, <https://www.eclipse.org/>, последно достъпен на 17.10.2021г.
- [16] “PhpStorm vs Eclipse”, https://www.slant.co/versus/812/1957/~phpstorm_vs_eclipse, последно достъпен на 17.10.2021г.
- [17] “PHP Tutorial”, <https://www.tutorialspoint.com/php/index.htm>, последно достъпен на 17.10.2021г.
- [18] “C# Tutorial”, <https://www.tutorialspoint.com/csharp/index.htm>, последно достъпен на 17.10.2021г.
- [19] “Python Tutorial”, <https://www.tutorialspoint.com/python/index.htm>, последно достъпен на 17.10.2021г.
- [20] “ASP.NET vs PHP: Which One To Choose and Why For App Development?”, <https://www.ongraph.com/asp-net-vs-php-which-one-to-choose/>, последно достъпен на 17.10.2021г.
- [21] “Python vs PHP”, <https://www.geeksforgeeks.org/python-vs-php/>, последно достъпен на 17.10.2021г.
- [22] “HTML Tutorial”, <https://www.tutorialspoint.com/html/index.htm>, последно достъпен на 17.10.2021г.
- [23] “CSS Tutorial”, <https://www.tutorialspoint.com/css/index.htm>, последно достъпен на 17.10.2021г.
- [24] “Javascript Tutorial”, <https://www.tutorialspoint.com/javascript/index.htm>, последно достъпен на 17.10.2021г.
- [25] “TypeScript Tutorial”, <https://www.tutorialspoint.com/typescript/index.htm>, последно достъпен на 17.10.2021г.
- [26] “The PHP Framework for Web Artisans”, <https://laravel.com/>, последно достъпен на 17.10.2021г.
- [27] “Symfony”, <https://symfony.com/>, последно достъпен на 17.10.2021г.
- [28] “Laravel vs Symfony in 2021 – Which PHP Framework Choose For Your Project?”, <https://asperbrothers.com/blog/laravel-vs-symfony/>, последно достъпен на 17.10.2021г.
- [29] “MongoDB vs MySQL”, <https://www.geeksforgeeks.org/mongodb-vs-mysql/>, последно достъпен на 17.10.2021г.
- [30] “MySQL Tutorial”, <https://www.mysqltutorial.org/>, последно достъпен на 17.10.2021г.

- [31] “N Tier(Multi-Tier), 3-Tier, 2-Tier Architecture with EXAMPLE”, <https://www.guru99.com/n-tier-architecture-system-concepts-tips.html>, последно достъпен на 17.10.2021г.
- [32] “Hypertext Transfer Protocol (HTTP)”, <https://www.extrahop.com/resources/protocols/http/>, последно достъпен на 17.10.2021г.
- [33] “What is a REST API?”, <https://www.redhat.com/en/topics/api/what-is-a-rest-api>, последно достъпен на 17.10.2021г.
- [34] “Blade Templates”, <https://laravel.com/docs/8.x/blade>, последно достъпен на 17.10.2021г.
- [35] “Artisan CLI”, <https://laravel.com/docs/5.0/artisan>, последно достъпен на 17.10.2021г.
- [36] “Eloquent ORM”, <https://laravel.com/docs/5.0/eloquent>, последно достъпен на 17.10.2021г.
- [37] “OOAD - UML Behavioural Diagrams”, https://www.tutorialspoint.com/object_oriented_analysis_design/ood_uml_behavioural_diagrams.htm, последно достъпен на 17.10.2021г.
- [38] Davis, Fred. “Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technolog” MIS Quarterly, Vol. 13, No. 3, 1989

**Публикувана научна статия
във връзка с дипломната работа**

Bontchev, B., Dankov, Y., Vassileva, D. and **Kovachev, M.** Software Instruments for Analysis and Visualization of Game-Based Learning Data, Proc. of 12th Int. Conf. on Applied Human Factors and Ergonomics (AHFE 2021), Manhattan, New York, USA, July 25-29, 2021, Advances in Intelligent Systems and Computing, Edited by Ahram, T.Z., Karwowski, W., and Kalra, J., LNNS, volume 271, Springer, ISSN: 21945357, 2021, pp.395-402 (SJR=0.184/Q3/2019)

**Използване на резултатите от дипломната работа в
научноизследователски проекти**

Резултатите от тази дипломна работа намират приложение в научноизследователския проект АПОГЕЙ, финансиран от Българския национален фонд за научни изследвания, съгл. Договор № DN12/7/2017.