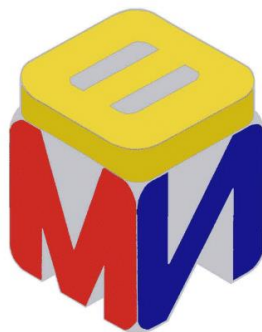


**П л о в д и в с к и   У н и в е р с и т е т**

**„ П а и с и й   Х и л е н д а р с к и ”**

---

**Факултет по математика и информатика**



**Дипломна работа**

**Forest Book**

**Образователна игра с потребителски интерфейс**

*Част I - потребителски интерфейс*

*Дипломант:*

Мария Атанасова

фак. № 1601417015

спец. Софтуерни Технологии

*Научен ръководител:*

ас.Александър Петров

кат. "Компютърни системи"

Пловдив 2018г.

# Съдържание

Списък на фигурите .....	4
Списък на използваните съкращения .....	6
Въведение .....	7
1.1. Основна цел на дипломната разработка .....	8
1.2. Съществуващи подобни игри .....	9
Глава 1 Архитектура и моделиране.....	10
1.1. Компоненти .....	10
1.1.1. Клиент .....	10
1.1.2. Сървър .....	10
1.1.3. База данни.....	11
1.1.4. Игра “Forest book” .....	11
1.2. Използвани технологии за разработка на проекта и техни алтернативи.....	12
1.2.1. React.js.....	12
1.2.2. Node.js сървър.....	19
1.2.3. Turn.js.....	20
1.2.4. Firebase .....	20

1.2.5. Unity .....	21
1.3. Инструменти и помощни средства за разработката .....	22
Глава 2 Реализация .....	23
2.1. Първи етап на разработка.....	23
2.1.1. Връзка между ГПИ и БД .....	25
2.1.2. Потребителски интерфейс .....	28
2.1.3. Резултат от първия етап на разработка .....	31
2.2. Втори етап на разработка.....	31
2.2.1. Начална страница.....	33
2.2.2. Административна форма .....	35
2.2.3. Промени в книгата с животни.....	36
2.2.4. Резултат от втория етап на разработка .....	36
Глава 3 Ръководство за потребителя .....	37
3.1. Начална екран .....	37
3.2. Логин .....	38
3.3. Регистрация .....	39
3.4. Меню на сайта .....	40
3.5. Относно играта (About).....	42
3.6. Статистика .....	42
3.7. "Book of animals" .....	43

3.8. Административен панел .....	44
Заклучение.....	45
Библиография.....	46

## Списък на фигурите

фигура 1 Методи на жизнения цикъл .....	14
фигура 2 Най-обичана рамка/библиотека за 2017 година .....	16
фигура 3 Angular е по-търсената библиотека за 2017г .....	16
фигура 4 Статистика на Npmcharts за тегленията на React и Angular .....	17
фигура 5 Разпределение на платформите според 60 000 оферти за работа .....	19
фигура 6 Top SDK Players (source: SafeDK Mobile Ltd.) .....	21
фигура 7 Статистика от 2014г за популярността на Unity .....	22
фигура 8 Диаграма на състоянията №1.....	24
фигура 9 Лого.....	29
фигура 10 Диаграма на състоянията №2.....	32
фигура 11 UseCase диаграма .....	32
фигура 12 Начален екран .....	37
фигура 13 Логин форма.....	38
фигура 14 Facebook popup window.....	39
фигура 15 Форма за регистрация.....	40
фигура 16 основна страница на сайта - меню .....	41
фигура 17 страница About .....	42

фигура 18 Таблица със статистика за играчите.....	43
фигура 19 Book of animals - flip book .....	43
фигура 20 Административен панел .....	44

## **Списък на използваните съкращения**

ГПИ - графичен потребителски интерфейс

БД – база данни

DOM – документен обектен модел (Document Object Model).

MVC - модел-изглед- контролер архитектурен модел

SPA - приложения от една страница (single-page applications).

CRUD - create, read, update, delete приложения

SDK - софтуерни инструменти за разработка (software development kit )

ГРС - гъвкава разработка на софтуер

## Въведение

Едва ли има човек, който не би оценил красотата на природата. Многообразието на прекрасни и вдъхновяващи гледки ни карат да затаим дъх пред величието им. Неразделна част от тази живописна картина са множеството видове животни, обитаващи Земята. Грациозността и очарованието на едни видове събуждат нашето възхищение, докато силата, бързината и непреклонността на други ни карат искрено да уважаваме животните, дори да изпитваме респект към тях.

Можем ли да си представим природата без животните? Какво би било да стъпиш в гората, а тя да не е огласена от песента на птиците, да го няма зайчето пребягващо през полянките, нито катеричката в клоните на дърветата. Една плашещо тиха природа, загубила живецата си. Дори във всекидневието си сме заобиколени с животни - те ни чакат с нетърпение вкъщи, разминаваме се с тях на път за работа. Всеки, имащ домашен любимец, може да каже каква радост, обич и нежност е внесло животинчето в живота му.

За жалост с годините, в следствие на дейността на хората, множество хабитати изчезват. Векове наред животните са избивани заради месото, кожата си, перата и по други подобни причини. В един момент дали заради човека или заради естественият подбор в природата някои от животните са на път да изчезнат като вид от нашата планета и страна. В последните години се забелязва и проявяване на особена жестокост към животните, живеещи в градска среда. С цел забава или поради безразличие все по-често кучета и котки биват изоставяни, осакатявани и дори убивани. Човека започва да губи връзката си с природата и животните в нея.



Темата за опазването на околната среда и защитата на животните набира все по-голяма популярност. Все повече хора се опитват да живеят без да използват пластмаса, рециклирайки по-голямата част от боклука си. Други водени от обичта си към животните избират да консумират месо, а в някои случаи и всякакви животински продукти. Организации като *PETA (People for the Ethical Treatment of Animals)* работят в посока защитата на правата на животните, спирането на жестокостите към тях и неизползването им в лаборатории. Организацията *Greenpeace* има за цел да гарантират способността на Земята да осигурява живота в цялото му разнообразие и фокусира усилията си към въпроси като глобалното затопляне, обезлесяването, свръхулова и промишления китолов.

Друг начин да се опази природното и животинско разнообразие е като още от ранна възраст децата биват възпитавани и учени на уважение и обич към природата и нейните обитатели. По един увлекателен и занимателен за тях начин трябва да им се представи живота и съдбата на различните видове животни, включително на изчезващи, застрашени или изчезнали вече биологични видове.

Разбира се това може да стане в училище, чрез разкази от родителите или като се прочете книга по въпроса. Какво е нещото обаче, което децата с радост вършат без да го приемат като задължение или с досада? Игрите винаги са имали предимство пред сухата теория. Какво по-хубаво от това да съчетаеш приятното с полезното? Да опознаваш животните докато играеш – това е същността на “Forest book”.

## **1.1. Основна цел на дипломната разработка**

Проекта “Forest book” цели създаването на *игра* с образователен характер, свързана със *сайт* с потребителски интерфейс и с централизирана *база данни*. Той е разработен съвместно от трима дипломанти, като трите обособени части ще са предмет на различни дипломни работи.

Цел на настоящата дипломна работа е създаването на сайт с лесен и удобен за използване графичен потребителски интерфейс, свързан към базата данни, който да бъде неразделна част от цялостния облик и функционалност на “Forest book”.

За постигането на тази цел се поставят следните основни задачи:

- Избор на функционалностите на сайта
- Избор на технологии
- Изграждане на отделните елементи
- Създаване на сайта
- Тестване на функционалностите

В глава първа, Архитектура и моделиране, е представена архитектурата на сайта, както и използваните технологии.

В глава втора, Реализация, е разгледан методът за реализация на сайта.

В глава трета, Ръководство за потребителя, е описана работата със сайта и графичния интерфейс.

## **1.2. Съществуващи подобни игри**

Съществуват множество игри базирани на животните като например: Animal Voyage: Island Adventure, Animal Rescue Book, Dr. Cares и други, като в тях образователният елемент е засегнат до различна степен.

Изброените по-горе игри имат сходна идея, но са за мобилни устройства. Игрите се харесват сред децата и възрастните, поради техния ненаатоварващ характер и красиви анимации.

Може да се допусне, че реализираният проект “Forest book” също би предизвикал интереса на потребителите на тези игри.

# Глава 1 Архитектура и моделиране

## 1.1. Компоненти

### 1.1.1. Клиент

С тази част от системата потребителите взаимодействат. Комуникацията със сървъра остава скрита. Графичния интерфейс отговаря за визуалното представяне на информацията, за оформлението на сайта, дава възможност на потребителите да влизат в сайта, както и да получават информация за постиженията си в играта. Също така ГПИ осигурява лесен и удобен за администраторите начин за въвеждане на нови животни в базата данни и извършването на корекции в нея.

Играта Forest book е достъпна единствено през персонален компютър вървящ под операционната система Windows.7/8/8.1/10.I. Въпреки това уеб сайта към играта е разработен с адаптивен дизайн, който позволява използването му както като десктоп, така и като мобилен сайт.

### 1.1.2. Сървър

Сървърната част осигурява връзката с базата данни, като обезпечава обработката на данни от базата. Тя предоставя бизнес логиката на системата. За момента се използва Node.js като localhost server.

### 1.1.3. База данни

Централизираната база данни съхранява необходимите на сайта и играта данни, свързани с регистрираните потребители, достигнатите от тях нива в играта, както и съответните им точки, информация за съществуващи и изчезнали животни и всякакви други необходими за правилното функциониране на системата данни. Базата данни реално осъществява връзката между Web частта и същинската игра, като позволява въвеждане на данни и от двете страни, както и визуализиране на въведените вече в базата данни. Благодарение на БД съхраняваната информация е синхронизирана и единна, като няма излишни повторения в нея.

Базата данни не е основна тема на тази дипломна работа. Тя се разработва от друг член на екипа и ще бъде представена в отделна дипломна работа. В настоящият труд БД ще се разгледа само до колкото се свързва с потребителския интерфейс.

### 1.1.4. Игра "Forest book"

"Forest book" предоставя 3D среда, в която се развива основната част на играта. Играчът изследовател има за задача да открие различни видове животни, които съществуват и в днешно време. При допир с тях те му се предлагат решаването на интересен 2D пъзел, решаването на който дават кратка информация за животното, с което се е срещнал играча като тези пъзели се използват за усложняване на играта и развиване на логическата мисъл на играча. В края на нивото се предоставя информация за вече изчезнал вид животно, което не може да се срещне в наше време.

Самата игра, нейното създаване и функционалност са тема на друга дипломна работа и няма да бъдат детайлно разглеждани в настоящата.

## 1.2. Използвани технологии за разработка на проекта и техни алтернативи

### 1.2.1. React.js<sup>1</sup>

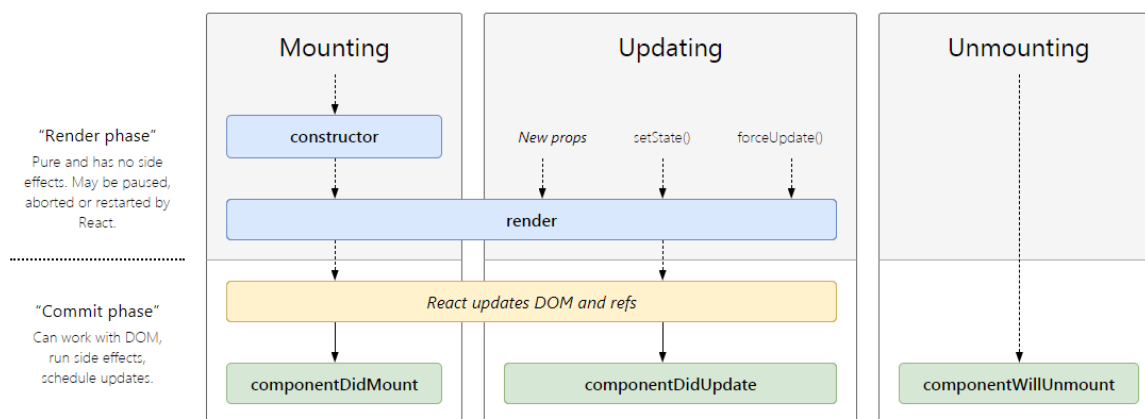
За създаването на Front-end частта автора използва **React.js** - една от най-популярните и бързи JavaScript библиотеки за разработване на клиентски интерфейс в света. Той има над 70 хиляди звезди в GitHub и над 1100 сътрудници. React позволява да се създават големи уеб приложения, които използват данни и могат да се променят във времето без презареждане на страницата, може да се използва като основа при разработването на едностранни или мобилни приложения. Неговата основна цел е да осигури скорост, простота и мащабируемост. Множество добре известни компании използват React.js библиотеката. Тя е разработвана и поддържана от Facebook и Instagram, използва се още от: Twitter, Amazon, Airbnb и др.<sup>2</sup>.

Компонентно базиран<sup>3</sup> – посредством React.js се изграждат капсулирани компоненти, които управляват собствените си състояния, и в следствие се композират, за да се създадат сложни потребителски интерфейси. Тъй като логиката на компонентите е написана на JavaScript вместо чрез шаблони, може лесно да се прехвърлят множество данни чрез приложението и да се запази състоянието извън DOM (Document Object Model).

Virtual Document Object Model е една от особеностите използвани в тази рамка. React създава кеш за структурата на данните в паметта, изчислява разликите получени в резултат на направена промяна и след това актуализира ефективно DOM. Това позволява писането на код, сякаш цялата страница се изобразява при всяка промяна, докато библиотеките React правят само елементите, които действително се променят.

**Жизнен цикъл на компонентите.** Компонентите имат няколко метода на жизнения цикъл, които могат да се променят, за да стартира кода в определено време от процеса. Методите разгледани по-долу обхващат най-често използваните при създаване на компоненти в React:

- **render()** – това е единственият задължителен метод в компонентния клас. Той трябва да остане “чист”, което означава, че не трябва да променя състоянието на компонентите, а да връща при всяко повикване един и същи резултат, като не си взаимодейства директно с браузъра.
- **constructor()** - обикновено в React конструкторите се използват само за две цели – за инициализиране на локалното състояние, чрез задаване на обект на `this.state` и за обвързване на методите за обработка на събития (`event handler methods`) с инстанция.
- **componentDidMount()** – този метод се вика след като компонента е създаден в потребителския интерфейс. Това обикновено се използва за задаване на заявка за зареждането на данни от отдалечен източник (например БД).
- **componentDidUpdate()** - извиква се след появата на актуализацията. Този метод не се извиква за първоначалното рендиране. Дава възможност за работа с DOM, когато компонентът е актуализиран.
- **componentWillUnmount()** – използва се непосредствено преди компонентът да бъде унищожен. Чрез него се извършват необходимите почиствания, като например обезсилване на таймерите, анулиране на заявки в мрежата или почистване на всички абонаменти, които са създадени от `componentDidMount()`.



фигура 1 Методи на жизнения цикъл

React прави лесно създаването на интерактивни потребителски интерфейси. Възможно е разработването на нови функции в React без да се пренаписва съществуващия код.

Алтернатива на React.js е платформата с отворен код за уеб приложения - **Angular.js**. Целта е да се опрости както разработването, така и тестването на такъв вид приложения, чрез предоставянето на платформа по модела на архитектурата - Модел-Изглед-Контролер, заедно с всички често използвани компоненти за интернет приложения<sup>4</sup>. Тя се поддържа от Google и общност от програмисти и корпорации, целящи справянето с различни трудности при разработването на приложения от една страница (single-page applications, или SPA).

AngularJS е разработен, като се следва идеята, че декларативното програмиране трябва да се използва за разработката на потребителски софтуер и свързвайки софтуерните компоненти, докато императивното програмиране е по-добре да се използва при дефинирането на бизнес логиката в приложението<sup>5</sup>. AngularJS не набляга толкова на DOM манипулацията, а цели подобряването на възможността за тестване на приложението и неговата производителност.

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

- Модел (Model) – основно включва информацията относно данните в приложението, която обикновено се взема от сървъра (Напр. име, презиме, фамилия, ЕГН, студентски номер, електронна поща и т.н.)
- Изглед (View) – представлява информацията, която се визуализира на екрана.
- Контролер (Controller) - представлява софтуерният код, който осъществява връзката между данните, които се пазят на сървъра (Модел) и това кои от тях да се визуализират на екрана (Изглед).

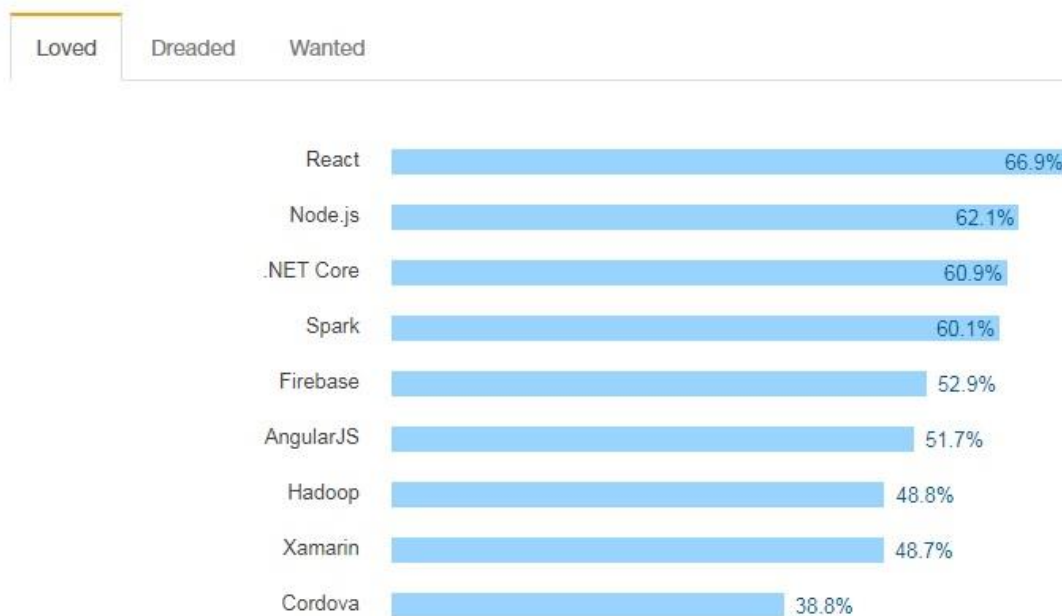
AngularJS опростява разработката на приложения, като представя на разработчика по-високо ниво на абстракция. Като всяка абстракция, тя идва с цената на гъвкавост. С други думи, не всяко приложение е подходящо за AngularJS. AngularJS е създаден с помощта на приложението CRUD. За щастие приложенията CRUD представляват по-голямата част от уеб приложенията.

Игрите и редакторите с графичен интерфейс са примери за приложения с интензивна и трудна манипулация в DOM. Тези видове приложения са различни от приложенията CRUD и в резултат на това вероятно не са подходящи за AngularJS. В тези случаи е по-добре да се използва библиотека с по-ниско ниво на абстракция, като jQuery.

Според **Stackoverflow Developer Survey 2017**<sup>6</sup> React е най-обичаната библиотека, но за същия период данните показват, че Angular е по-желаната от двете:

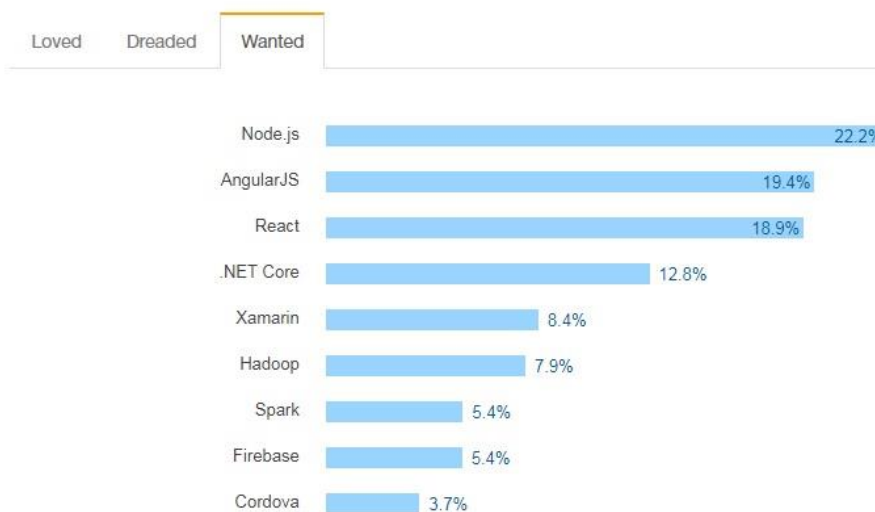


## Most Loved, Dreaded, and Wanted Frameworks, Libraries and Other Technologies



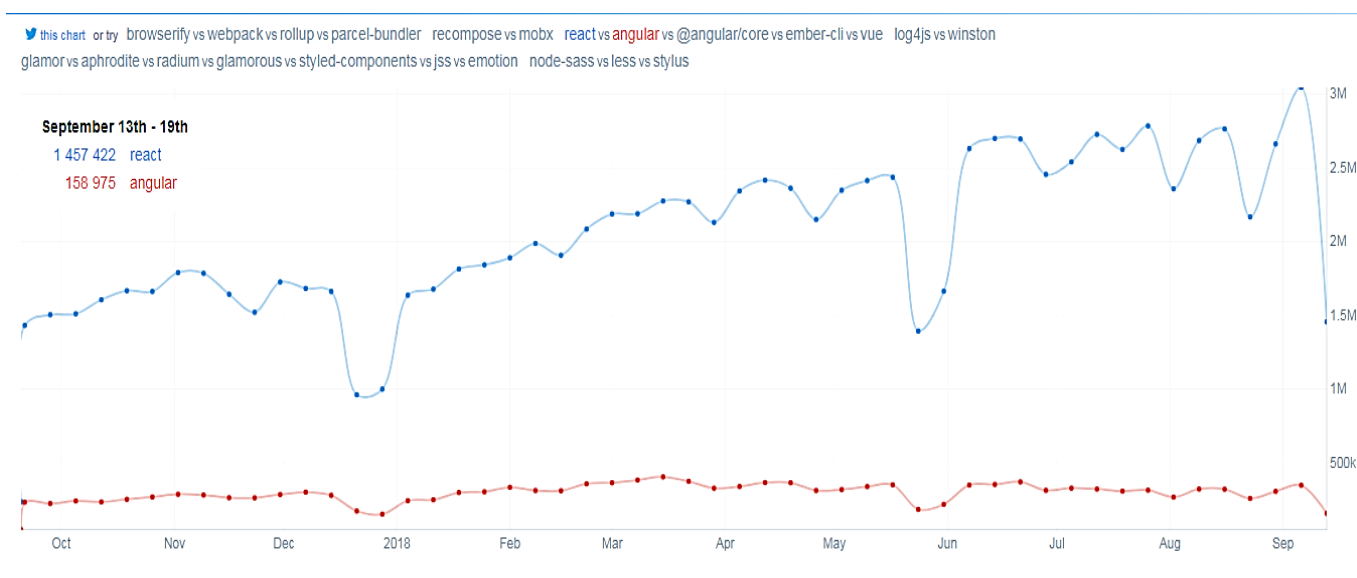
фигура 2 Най-обичана рамка/библиотека за 2017 година

## Most Loved, Dreaded, and Wanted Frameworks, Libraries and Other Technologies



фигура 3 Angular е по-търсената библиотека за 2017г

На представената по-долу фигурата е показана статистика на Npmcharts за тегленето на React и Angular:



фигура 4 Статистика на Npmcharts за тегленията на React и Angular

Въпреки горепосочената графика, данните сочат, че Angular се използва широко от Google и др. компании, а корпоративните софтуерни проекти често използват корпоративен регистър за софтуерни артефакти. Ако Angular е по-популярен в корпоративните среди, това може да обясни по-малкото изтегляния от публичния регистър npm.

В последно време още една рамка доби популярност.

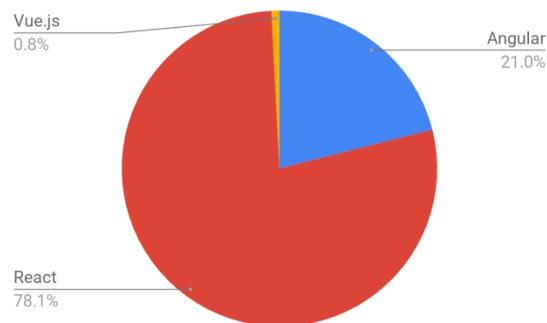
**Vue.js** <sup>7</sup>е JavaScript рамка с отворен код, която идеално се вписва в създаването на високо адаптивни потребителски интерфейси и сложни приложения от една страница. Тя има своите предимства и недостатъци.

### Предимства на Vue.js:

- Vue.js има много подобни характеристики с Angular и това може да помогне за оптимизирането на обработката на HTML блокове с използването на различни компоненти.
- Подробна документация. Vue.js има много подробна документация, която може да улесни обучението на разработчиците и да спести много време, при разработване на приложение, използвайки само основните познания за HTML и JavaScript.
- Адаптивност. Той осигурява бърз преход от други рамки към Vue.js поради сходството с Angular and React от гледна точка на дизайна и архитектурата.
- Голямо мащабиране. С vue.js могат да се разработят доста големи шаблони за многократна употреба, които да бъдат направени бързо, поради простата структура на рамката.
- Малък размер. Vue.js може да е с размер от около 20 KB, като запази скоростта и гъвкавостта си, което позволява постигането на много по-добри резултати в сравнение с други рамки.

### Недостатъците на Vue.js:

- Липса на ресурси. Vue.js все още има доста малък пазарен дял в сравнение с React или Angular, което означава, че споделянето на знания в тази рамка е все още в началната фаза.
- Риск от прекалена гъвкавост. Понякога Vue.js може да има проблеми, докато се интегрира в огромни проекти, защото все още няма много натрупан опит с възможни решения.



фигура 5 Разпределение на платформите според 60 000 оферти за работа

Както се вижда от проучванията и трите рамки имат своите потребители. Те предоставят добри възможности за изграждането на уебсайтове и до известна степен е въпрос на предпочитание, коя ще бъде използвана. Тъй като по време на курса на обучение библиотеката React.js бе предмет на по-подробно разглеждане, автора се е спрял именно на нея за разработването на website към играта.

### 1.2.2. Node.js сървър<sup>8</sup>

Node.js предоставя среда за изпълнение на JavaScript приложения на сървъра. Той е мултиплатформен с отворен код, използва се на OS X, Microsoft Windows, Linux. Разработен е върху двигателя V8 на Google (виртуална машина за изпълнение на JavaScript). Предоставя задвижвана от събития архитектура и неблокираща входно-изходна система за програмиране на приложенията. Целта на Node.js е да предостави лесен начин за изграждане на мащабируеми мрежови програми.

Node е еднонишков и не изключва ресурси. Той има една нишка на процес и поради това, е невъзможно няколко нишки да правят промени по едни и същи данни едновременно. Тъй като нищо не блокира, Node позволява да бъде използван от неексперти програмисти за разработване на бързи системи.

Скоростното изпълнение се дължи на V8, който компилира JavaScript в native машинен код, вместо да го интерпретира или да го изпълнява като байткод.

Node.js съдържа вградени библиотеки, които позволяват приложенията да работят като сървъри, без софтуер като Apache HTTP Server или IIS. Node придобива популярност като сървърна платформа и се използва от Microsoft, Yahoo!, Walmart, Groupon, SAP, LinkedIn, Rakuten, PayPal, Voxer и GoDaddy.

### 1.2.3. Turn.js

Turn.js е JavaScript библиотека, jQuery плъгин, която има възможността да направи съдържанието на страницата да изглежда като истинска книга или списание, използвайки всички предимства на HTML5. Притежава разработени множество детайли като сенки, светлосенки, опции за реалистично отгръщане на меки и твърди страници и др. Сайта придобива красив вид с новия потребителски интерфейс базиран на HTML5. Turn.js е най-добър за създаването на списания, книги и каталози.<sup>9</sup>

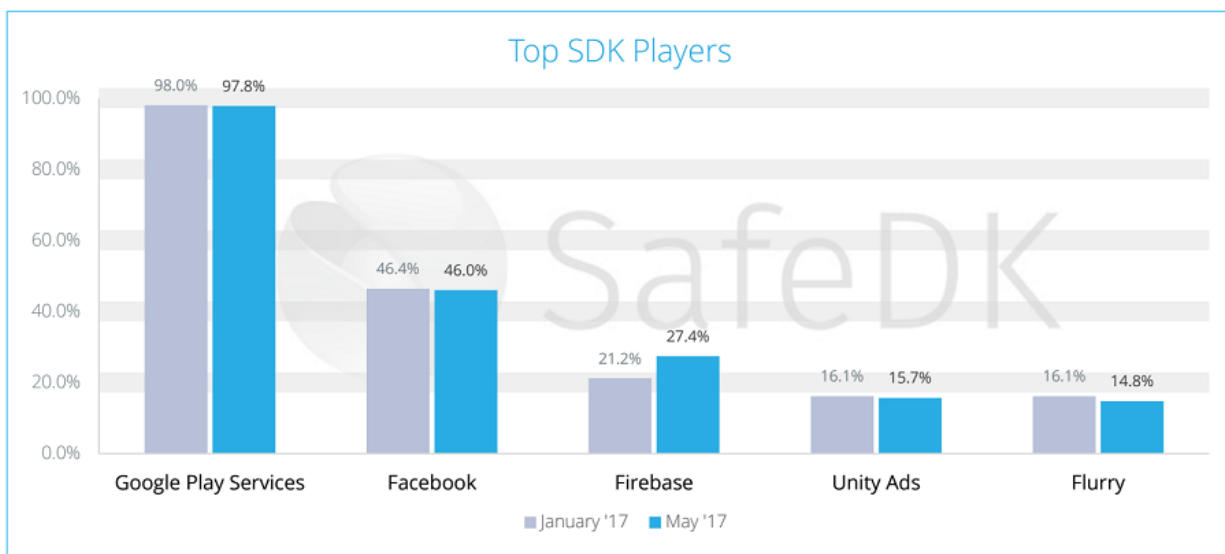
Библиотеката е използвана в реализацията на проекта, като чрез нея се визуализира своеобразна книга (flip book), в която всеки потребител може да открие животните, за които е събрал вече информация от играта. Turn.js предава на тази част от сайта един приятен завършек, допринасящ за цялостният замисъл и дизайн, както на сайта, така и на играта.

### 1.2.4. Firebase

За разработването на базата данни, съхраняваща нужната на играта и сайта информация, е избран Firebase.

Firebase е платформа за разработка на мобилни и уеб приложения, разработена от Firebase и в последствие придобита от Google. Масивният комплект за разработка на приложения (software development kit – SDK) на Firebase има много компоненти, към които Google добавя данни от години. В края на 2016 г. по-малко от 400 000 приложения използват главния комплект за разработване на софтуер Firebase. Днес, според MightySignal, почти 840 000 приложения имат инсталиран този SDK.<sup>10</sup>

От графиката е видно, че в периода от Януари до Май 2017г. Firebase SDK има най-голям растеж на популярността си<sup>11</sup>:



фигура 6 Top SDK Players (source: SafeDK Mobile Ltd.)

Както бе споменато, базата данни и по-подробното разглеждане на платформата Firebase са предмет на друга дипломна работа.

### 1.2.5. Unity

Самата игра е реализирана чрез междуплатформения игрови двигател Unity, който се използва за разработката на видео игри за компютри, конзоли мобилни устройства както и за сайтове.

Понастоящем 34% от първите 1000 безплатни мобилни игри се правят с **Unity**. Това е платформа за артисти, дизайнери и разработчици, които създават и си сътрудничат, със зашеметяващо кинематографично съдържание и геймплей последователности, като използват 2D и 3D дизайнерски инструменти, бърз режим за редактиране и итерация и мощна анимационна система.

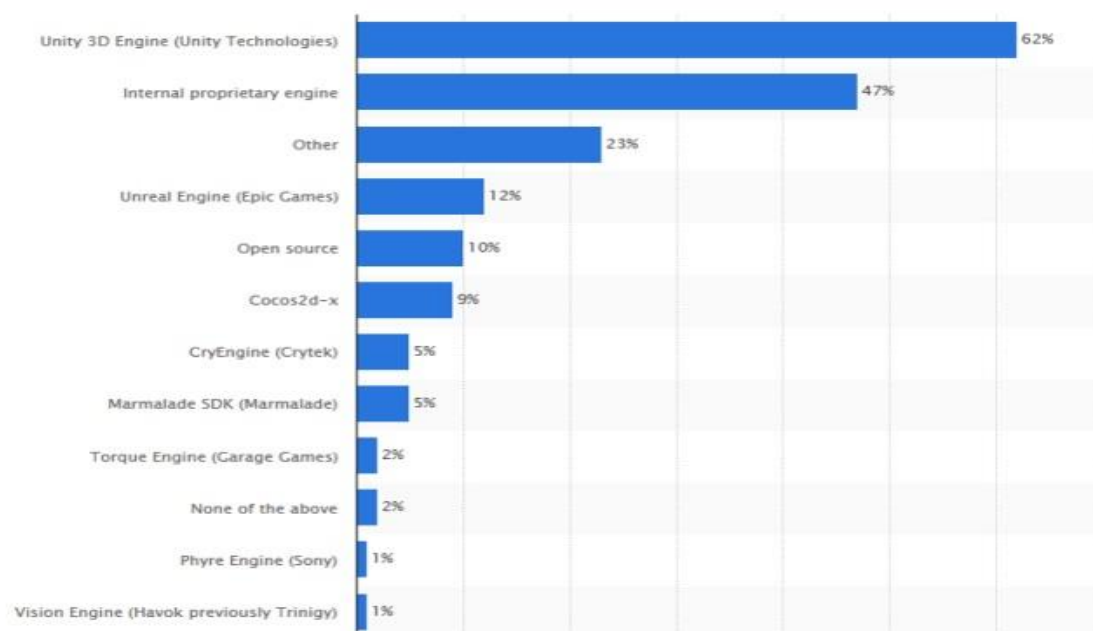
**Unreal** е друга алтернатива за разработване на игри. Това е може би един от най-популярните игрови двигатели в света и "най-успешният двигател на видеоигрите", награден от Guinness World Records. Това определено е най-добрият двигател, ако се работи върху голяма, сложна игра и е нужен силен 3D гейм енджин. Недостатък е обаче, че размерът на игрите ще е по-голям и ще трябва да се използват устройства от висок клас, за да се стартират.

Докато **Unreal** може да бъде по-популярен с компютърни и конзолни игри, **Unity** определено е по-предпочитан в мобилните игри и се е превърнал в геймърски двигател за много разработчици на мобилни игри.<sup>12</sup>

Според статистиката на сайта Statista за 2014г Unity заема водещо място сред най -популярните игровите двигатели във Великобритания:

#### What game engines do you currently use?

This statistic illustrates the most popular game engines used for video game development in the United Kingdom (UK) in 2014. The greatest share of respondents reported using Unity 3D Engine, at 62 percent.



фигура 7 Статистика от 2014г за популярността на Unity

### 1.3. Инструменти и помощни средства за разработката

Инструменти:

- WebStorm за Windows
- Adobe Photoshop CS6

Помощни средства:

- Github

## Глава 2 Реализация

За реализирането на сайта към играта Forest book има възможност да се подходи по различни начини, чрез използването на различни методи за разработка. Тъй като настоящата дипломна работа е замислена като продължение и естествено надграждане на курсовия проекта, разработен и представен на два пъти в дисциплините “Практически проект № 1” и “Практически проект №2”, е уместно да се допусне, че ще се наложат чести промени и добавяне на нови функционалности. В същото време е нужно грешките да се откриват във възможно най - ранен етап и да бъдат отстранявани своевременно.

Поради тези изисквания подходящо е да се използва гъвкавата разработка на софтуер (ГРС), която предлага и множество методи за разработка.

### 2.1. Първи етап на разработка

Първият етап на разработката включва избирането на технологии и създаването на основната архитектура на сайта.

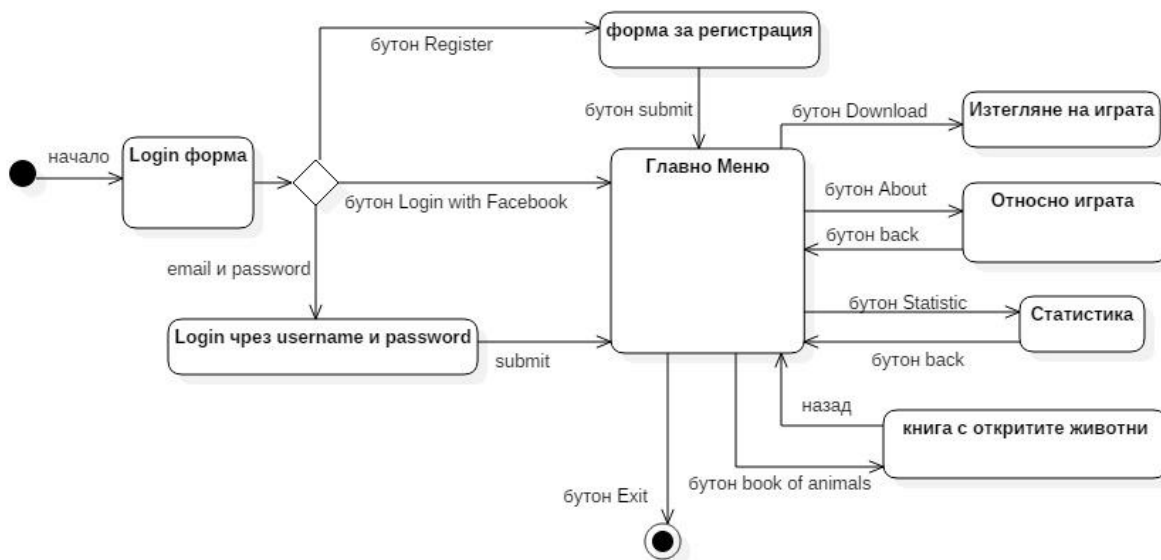
Този етап има за цел да изгради и постигне следните функционалности и задачи:

- Логин форма и форма за регистрация на нови потребители



- Създаване на меню, осигуряващо достъп на потребителя до всички функционалности
- Статистика на играчите
- Страница “About”
- Книга с животни “Book of animals”
- Свързване на графичния интерфейс с back-end частта

Представената по-долу фигура показва диаграмата на състоянията на сайта, върху която е изграден този етап от проекта.



фигура 8 Диаграма на състоянията №1

### 2.1.1. Връзка между ГПИ и БД

Комуникацията между базата данни и сайта се постига чрез използването на функционалностите на Firebase, като в йерархията на сайта бяха създадени следните js файлове :

- firebase.js – тук се инициализира и конфигурира firebase
- db.js – интерфейс между Firebase и сайта за ползване на базата данни
- auth.js - дефинира API за удостоверяване на Firebase, за да се ползват възможностите за регистрация, влизане и излизане на потребителите в сайта. Това е интерфейсът между официалното API на Firebase и сайта.
- index.js - файл за влизане в Firebase модула, чрез групиране и показване на всички модули изброени по-горе в един файл. Поради това не е необходимо другите модули в приложението да имат достъп до друг файл, освен този, за да използват неговите функционалности.

Същинската връзка с Firebase се инициализира във файла firebase.js

```
import * as firebase from 'firebase';
import '@firebase/firestore'

const config = {
  apiKey: "AIzaSyDs1H02QzfxDwxPkTOKvINpgjauwf1nh00",
  authDomain: "forest-book.firebaseio.com",
  databaseURL: "https://forest-book.firebaseio.com",
  projectId: "forest-book",
  storageBucket: "forest-book.appspot.com",
  messagingSenderId: "315342178024"
};

require("firebase/app");

require("firebase/auth");
require("firebase/database");

const app = firebase.initializeApp(config);
const db = firebase.firestore();
const auth = firebase.auth();
const facebookProvider = new firebase.auth.FacebookAuthProvider()
export { app, db, auth, facebookProvider };
```

Импортиран е firebase модула и е инициализиран софтуерния инструмент за разработка (SDK). Достъпни са Firebase услугите, които ще се използват в последствие в различните елементи на сайта. Класа FirebaseAuthProvider се използва, за да се осигури възможност за автентификация, чрез попън прозорец за влизане във Facebook. За това служи authWithFacebook компонента.

```
authWithFacebook() {  
  app.auth().signInWithPopup(facebookProvider).then(  
    (result, error) => {  
      if (error) {  
        this.toaster.show({intent: Intent.DANGER, message:  
          "Unable to sign in with Facebook" })  
      } else {  
        this.setState({redirect: true})  
      }  
    }  
  )  
}
```

В сайта е предвидена възможност за логин не само чрез фейсбук, но също така и чрез имейл и парола за вече регистрирани потребители. На показания по-долу код се вижда също така и че се прави проверка за определен предварително зададен административен имейл, която проверка в последствие препраща към административния панел посредством Routes.

Тази проверка както и цялата административна част на графичния интерфейс бе разработена на втория етап на разработката.

```

auth.doSignInWithEmailAndPassword(email, password)
  .then((result) => {
    this.setState(() => ({ ...INITIAL_STATE }));

    let afterLogin = Routes.MENU;
    if(result.email === 'admin@abv.bg'){
      afterLogin = Routes.ADMIN
    }
    history.push(afterLogin);
  })
  .catch(error => {
    this.setState(byPropKey('error', error));
  });

```

Формата за регистрация създава навия потребител, като запазва данните му в Authentication таблицата във Firebase и в същото време ги изпраща към Realtime Database на Firebase.

```

auth.doCreateUserWithEmailAndPassword(email, passwordOne)
  .then(authUser => {
    this.setState(() => ({ ...INITIAL_STATE }));

    const rootRef = _firebase.database();
    const post = rootRef.ref().child('users').child(authUser.uid);

    post.child("email").set(email);
    post.child("name").set(name);
    post.child("username").set(username);

    history.push(Routes.MENU);
  })
  .catch(error => {
    this.setState(byPropKey('error', error));
  });

```

При промяна в полета за въвеждане, в метода за рендиране на компонента, те актуализират локалното състояние на компонента, като използват onChange handler (такъв пример е даден в следващия фрагмент от кода).

В метода setState () се използва функция byPropKey (), която взема ключовата стойност (key value) и действителната стойност, въведена в полето. Във функцията byPropKey () ключовата стойност на полето се използва като динамичен ключ за разпределяне на въведения текст в local state object.

```
const {
  name,
  username,
  email,
  passwordOne,
  passwordTwo,
  error,
} = this.state;

const isValid =
  passwordOne !== passwordTwo ||
  passwordOne === '' ||
  email === '' ||
  username === '';

return (
  <form onSubmit={this.onSubmit}>
    <div className="row">
      <div className="col-lg-3"/>
      <div className="col-lg-6">
        <div className="screenreg">
          <div className="row">
            <header/>
            <div className="col-lg-3"/>
            <div className="col-lg-6">
              <input className="inputform"
                value={this.state.name}
                onChange={event => this.setState(byPropKey('name', event.target.value))}
                type="text"
                placeholder="Enter your Full Name"
              />
            </div>
          </div>
        </div>
      </div>
    </div>
  </form>
)
```

## 2.1.2. Потребителски интерфейс

Потребителският интерфейс има адаптивен дизайн, който позволява приятния му вид както на десктоп, така и на по - малък екран. В първия етап бяха използвани Material-ui компоненти, които чрез ефектите си спомогнаха за един изчистен и интересен вид на формите за регистрация и логин (подчертаване на полето, на което се намиращ, указателен текст в hintText, както и floatingLabelText).

В последствие бе предпочетено да не се използват Material-ui компонентите, а да се персонализира дизайна на бутоните и полетата за писане, за да съответстват по-точно на идеята на проекта.

```
<input className="inputform"
  value={passwordTwo}
  onChange={event => this.setState(byPropKey('passwordTwo', event.target.value))}
  type="password"
  placeholder="Confirm Password"
/>
<br/>
<button className="button"
  disabled={isInvalid}
  type="submit">
  <img src={require("./woodsign.png")}
    width="150"
    height="80"
  >
  </img>
  <div className="centered">Submit</div>
</button>
```

За дизайна на сайта автора се е придържал към първоначалната идея, описана във Въведението. Всички графични елементи са издържани в стил близък до природата (бутоните например са направени да изглеждат като дървени табели).

За завършения вид на Сайта допринеса и логото на играта. То е създадено специално за нуждите на проекта. Изработено е посредством Adobe Photoshop и възпроизвежда в себе си идеята на заглавието.



фигура 9 Лого

Интересен момент от реализацията на интерфейса представлява книгата със събраните от играчите животни. Първоначално бе направен опит книгата да се изгради въз основа на пакетите react-flip-book и react-flip-page. Резултата от графиката и анимацията на книгата, обаче не беше задоволителен. Възможно е и поради не голям опит в работата с тези пакети, автора да не е съумял да предаде по-реалистичен облик на създадения flip book.

Поради тези причини за реализацията на тази част от ГПИ бе използвана библиотеката Turn.js. Работата с библиотеката е сравнително лесна за изучаване, а резултата напълно задоволява изискванията към книгата. Представен е част от кода, използван при изграждането на книгата.

```
<div id="container" style="position:absolute;">
  <div id="flipbook" style="...">
    <div style="background-image: url('img/bookcover.jpg'); background-size: 100% 100%; " class=" hard"> </div>
    <div style="background-image: url('img/pagef.jpg'); background-size: 100% 100%; " class="hard"></div>
    <div style="background-image: url('img/pagef.jpg');background-size: 100% 100%;"...>
    <div style="background-image: url('img/pagef.jpg');background-size: 100% 100%;"...>
    <div style="background-image: url('img/pagef.jpg');background-size: 100% 100%;"...>
    <div style="background-image: url('img/pagef.jpg');background-size: 100% 100%;"...>
    <div style="background-image: url('img/pagef.jpg');background-size: 100% 100%;"...>
    <div style="background-image: url('img/pagef.jpg');background-size: 100% 100%;"...>
    <div style="background-image: url('img/pagef.jpg');background-size: 100% 100%;"...>
    <div style="background-image: url('img/pagef.jpg');background-size: 100% 100%;">6</div>
    <div style="background-image: url('img/pagef.jpg');background-size: 100% 100%; " class="hard"></div>
    <div style="background-image: url('img/bookcover.jpg');background-size: 100% 100%; " class=" hard"></div>
  </div>
</div>
<script type="text/javascript">
  $(function() {
    $("#flipbook").turn({
      width: 900,
      height: 550,
      autoCenter: true,
    });
  });
</script>
```

Недостатък, що се отнася до направения flip book, е че тъй като той е изграден на базата на HTML не може лесно да се имплементира към останалата част от графичния интерфейс разработена на React.

```

<a className="ButtonLink"
  href={'http://localhost:63342/reactmaster/src/components/login/Flipbookpage.html?_ijt=hqf4m1qb4tsqb5j4kv95mg0h68'}>

  <button className="button">
    <img src={require("./woodsign.png")}
      width="230"
      height="80"
      onClick={(event) => this.showBook(event)}>
    </img>
    <div className="centered">Book of animals</div>
  </button>
</a>

```

За да се препрати към книгата, след натискане на бутона се използва линк към html страницата. Това не е препоръчително да се прави в React. Следва да се намери начин за безопасното преминаване от javascript към html страниците.

### 2.1.3. Резултат от първия етап на разработка

След приключване на първия етап на разработка имаме работещ сайт, покриващ изискванията поставени в началото.

## 2.2. Втори етап на разработка

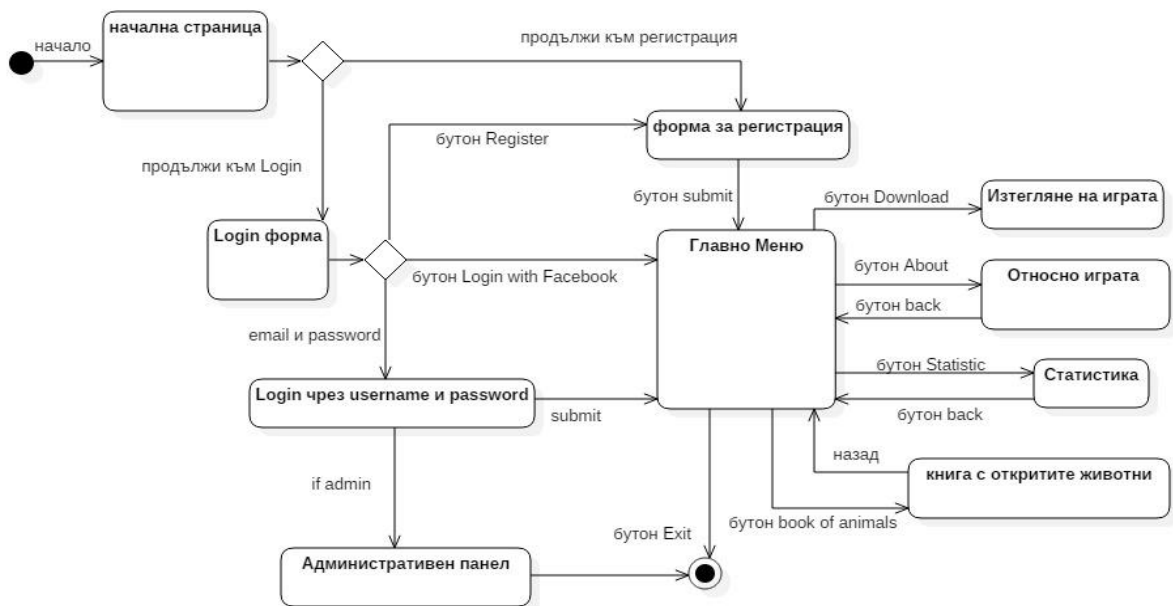
След създаването на основните функционалности на проекта в първия етап на разработката, тук бяха добавени нови такива и направени известни промени по дизайна на графичния интерфейс.

В този етап бе решено да се имплементират следните функционалности:

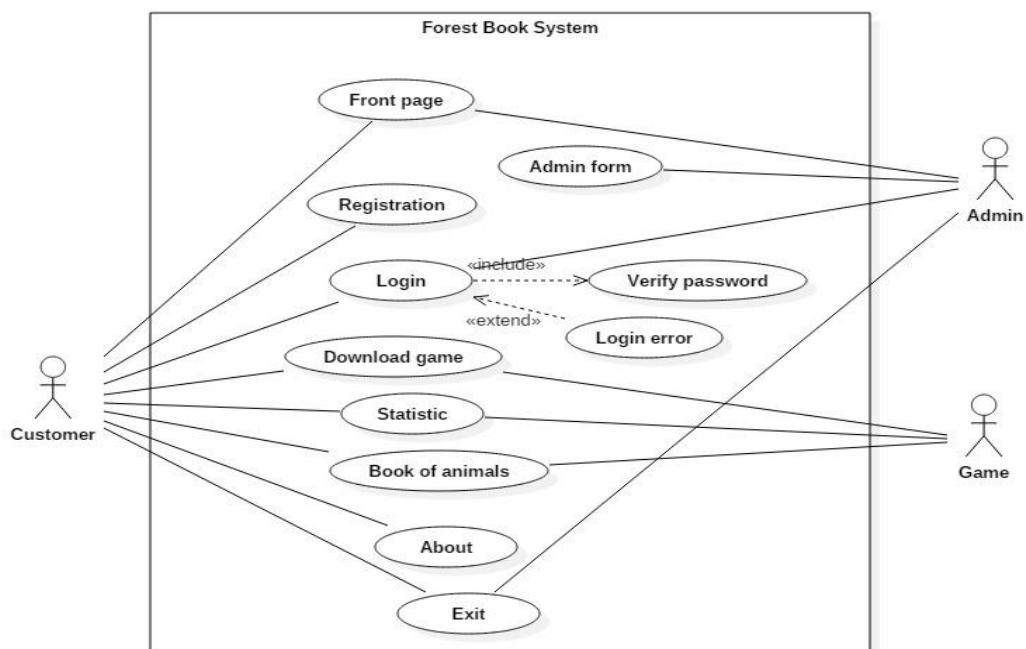
- Създаване на нова начална страница
- Административна форма
- Създаване на проверка за административен имейл
- Въвеждане на нови животни през админ формата
- Бяха направени някои промени в дизайна на сайта
- Промяна във книгата с животни, с цел достъпа и до базата данни

Диаграмата на състоянията показва как трябва интерфейса да обработва заявките на крайния потребител. Показана е и UseCase диаграма на сайта.





фигура 10 Диаграма на състоянията №2



фигура 11 UseCase диаграма

## 2.2.1. Начална страница

В първия етап за начална страница бе използвана тази за Логин. Създадена е самостоятелна начална страница представлява играта. За направление към Логин страницата и тази за регистрация се ползва navigation bar.

```
<nav className="Header" >
  <div className="row">
    <div className="col-md-8">

    </div>
    <div className="col-lg-4">
      <RaisedButton
        backgroundColor={black}
        label="Login"
        key="Login"
        style={styles.button}
        labelColor={styles.linkColor}
        containerElement={<Link className="HeaderLink" to="/login">Login</Link>}
        icon={<span className="glyphicon glyphicon-list-alt" aria-hidden="true"></span>}
      />
      <RaisedButton
        backgroundColor={black}
        label="Registration"
        key="Registration"
        style={styles.button}
        labelColor={styles.linkColor}
        containerElement={<Link className="HeaderLink" to="/registration">Registration</Link>}
        icon={<span className="glyphicon glyphicon-align-justify" aria-hidden="true"></span>}
      />
    </div>
  </div>
</nav>
```

Навигацията е разписана във файла Navigation.js и може да се зададе използването му чрез App.js

```
const routes = [
  {
    path: '/',
    exact: true,
    header: () => <Navigation/>,
    middle: () => <Frontpage/>
  },
  { "path": '/login'... },
  { "path": '/menu'... },
  { "path": '/registration'... },
  { "path": '/statistic'... },
  { "path": '/about'... },
  { "path": '/admin'... }
]
```

Компонентите са обвити в компонента Router (маршрутизатор), осигурен от React Router. Маршрутизаторът дава възможност за навигация от URL към URL в приложението на клиента, без да се правят заявки към уеб сървър. По този начин заявките трябва да бъдат поискани само веднъж от сървъра, като процесът на обработка на заявените маршрути се обработва от страна на клиента.

```
<Router>
  <div className="container">
    <div className="row">
      <div className="col-md-12">
        {routes.map((route, i) =>
          <Route
            key={i}
            path={route.path}
            exact={route.exact}
            component={route.header}
          />
        )}
      </div>
    </div>
    <div className="row">
      <div/>
      <div/>
      <div className="col-md-12">
        {routes.map((route, i) =>
          <Route
            key={i}
            path={route.path}
            exact={route.exact}
            component={route.middle}
          />
        )}
      </div>
    </div>
  </div>
</Router>
```

### 2.2.2. Административна форма

Основното надграждане в този етап на разработка е свързано със създаването на административна форма, която да позволява лесен достъп до съдържанието на базата данни.

След осъществяване на проверката разгледана в секция 2.1.1 “Връзка между ГПИ и БД” на администратора се предоставя достъп до административната страница.

Чрез достъпване на Firebase услугата database показаният код пушва името, снимка и описанието на животното, като ги прави атрибути на полета animals. Всяка нова заявка за писане получава уникален ключ автоматично.

```
56
57
58     handleSubmit(event) {
59         const rootRef = firebase.database();
60         const post = rootRef.ref('animals');
61         var data = {
62             name: this.state.name,
63             description: this.state.description,
64             img: this.state.ime,
65         }
66         post.push(data);
67         event.preventDefault()
68     }
69
70
```

След въвеждане в БД информацията се визуализира на екрана. За тази цел се използва componentDidMount метода. Достъпват се child полетата на базата и се извличат стойностите на атрибутите.

```

72
73     componentDidMount() {
74
75         const rootRef = firebase.database().ref();
76         const post = rootRef.child('animals').orderByKey();
77
78
79         post.once('value', snap => {
80             snap.forEach(child => {
81                 this.setState({
82                     id: this.state.id.concat([child.key]),
83                     nameArr: this.state.nameArr.concat([child.val().name]),
84                     img: this.state.img.concat([child.val().img]),
85                     descriptionArr: this.state.descriptionArr.concat([child.val().description])
86                 });
87             });
88         });
89     }
90 }

```

### 2.2.3. Промени в книгата с животни

Чрез имплементиране на връзка с базата данни, бе постигнато достъпването на данните и визуализирането единствено на записаните към полетата на потребителя животни.

По този начина книгата се персонализира за всеки играч.

### 2.2.4. Резултат от втория етап на разработка

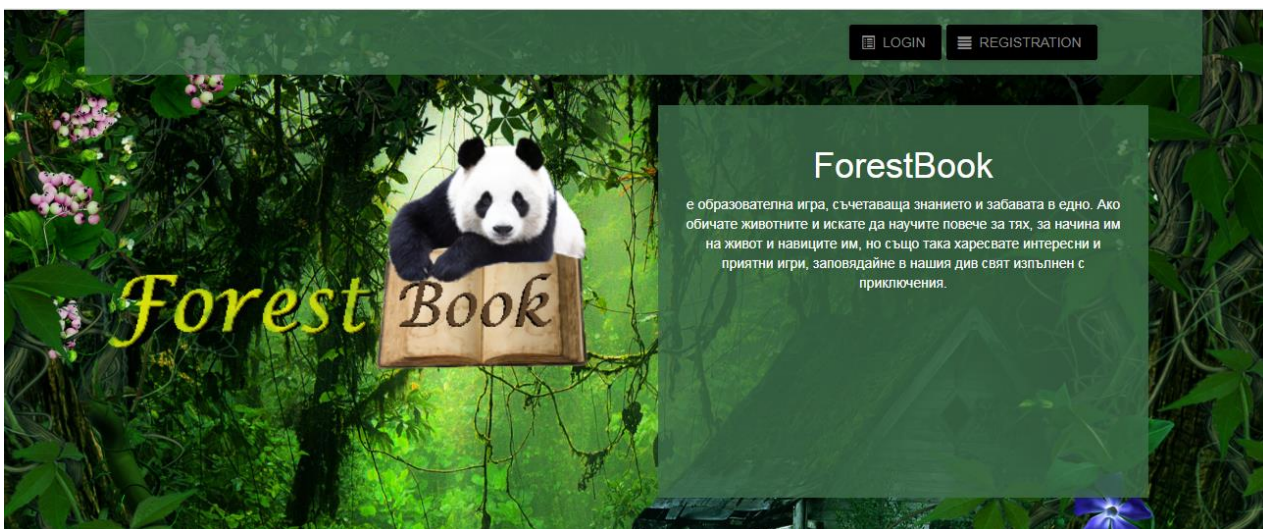
Задачите поставени в началото на вторият етап на развитие са постигнати. Администратора има достъп до базата данни, използвайки графичния интерфейс.

## Глава 3 Ръководство за потребителя

В тази глава чрез екранни снимки са представени основните функционалности на сайта и тяхното използване.

### 3.1. Начална екран

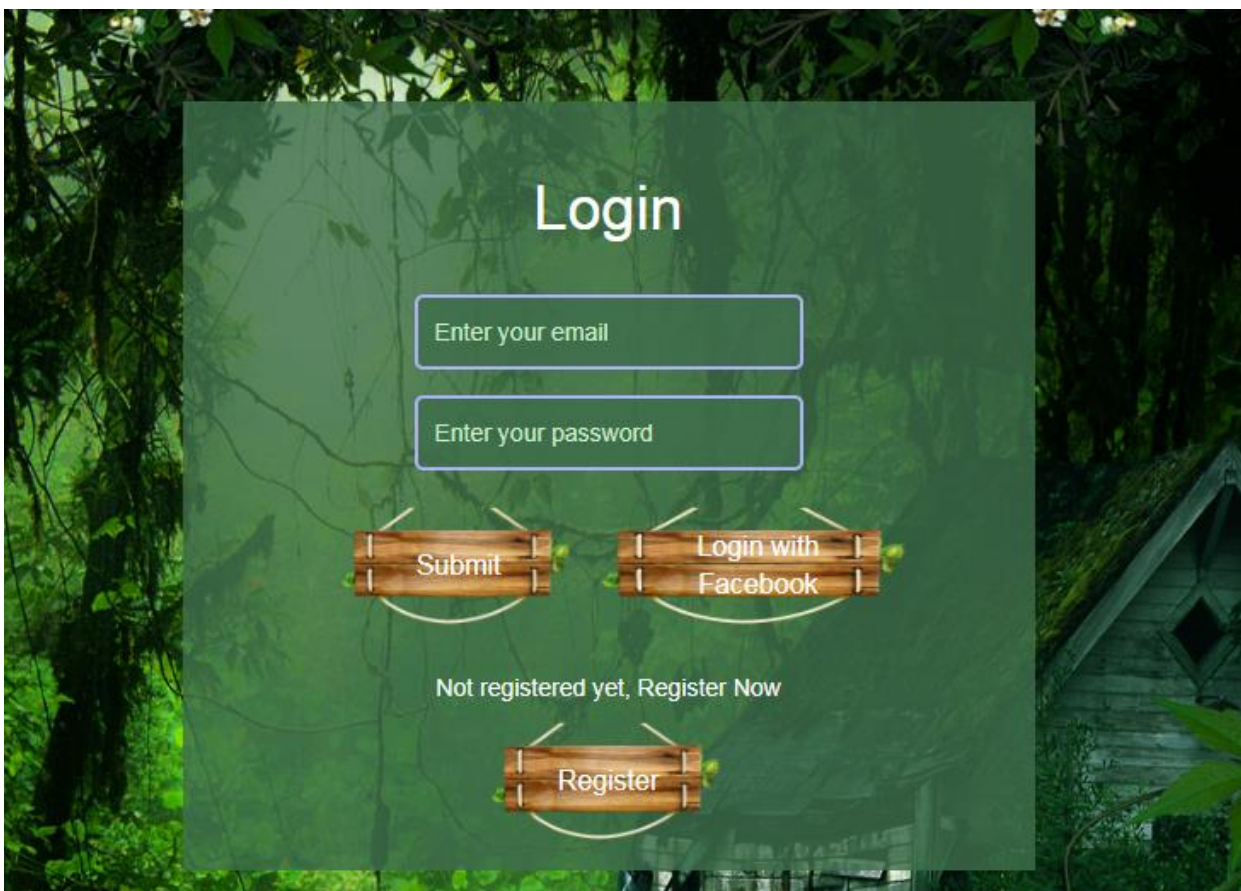
Стартирането на сайта отваря началната страница, която представя накратко целта на Forest book и предоставя възможност за регистрация на нов потребител или за “Login”, под формата на бутони, намиращи се в навигационен панел.



фигура 12 Начален екран

## 3.2. Логин

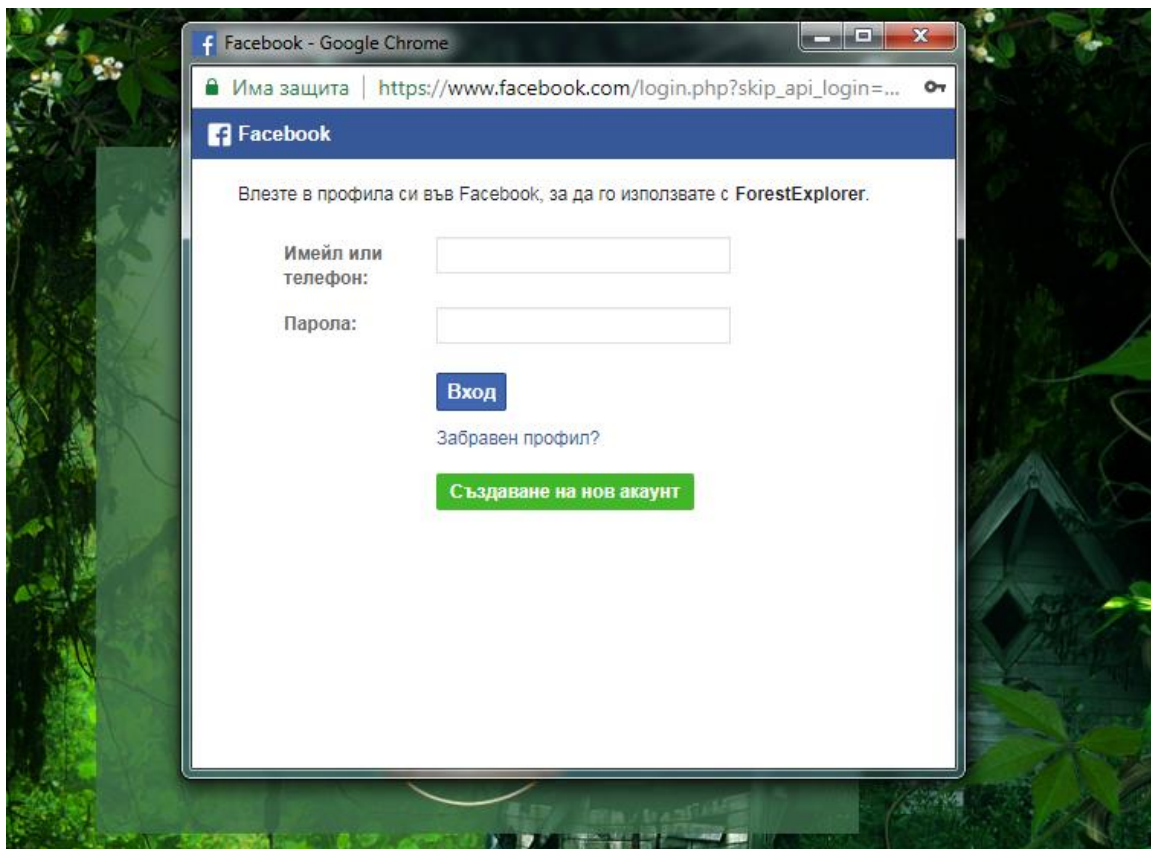
Избирайки възможността “Login” се визуализира Логин формата, чрез която потребителят може, след като се аутентикира, да влезе в основното меню. Предоставя се възможност за влизане чрез Facebook акаунт или чрез имейл и парола, ако потребителя е вече регистриран и фигурира в базата данни.



фигура 13 Логин форма

Ако потребителя желае да използва Facebook акаунта си и вече се е аутентикирал там през същото устройство, няма да е нужно повторно въвеждане на имейл и парола. В противен случай, му се отваря следната рорир форма.



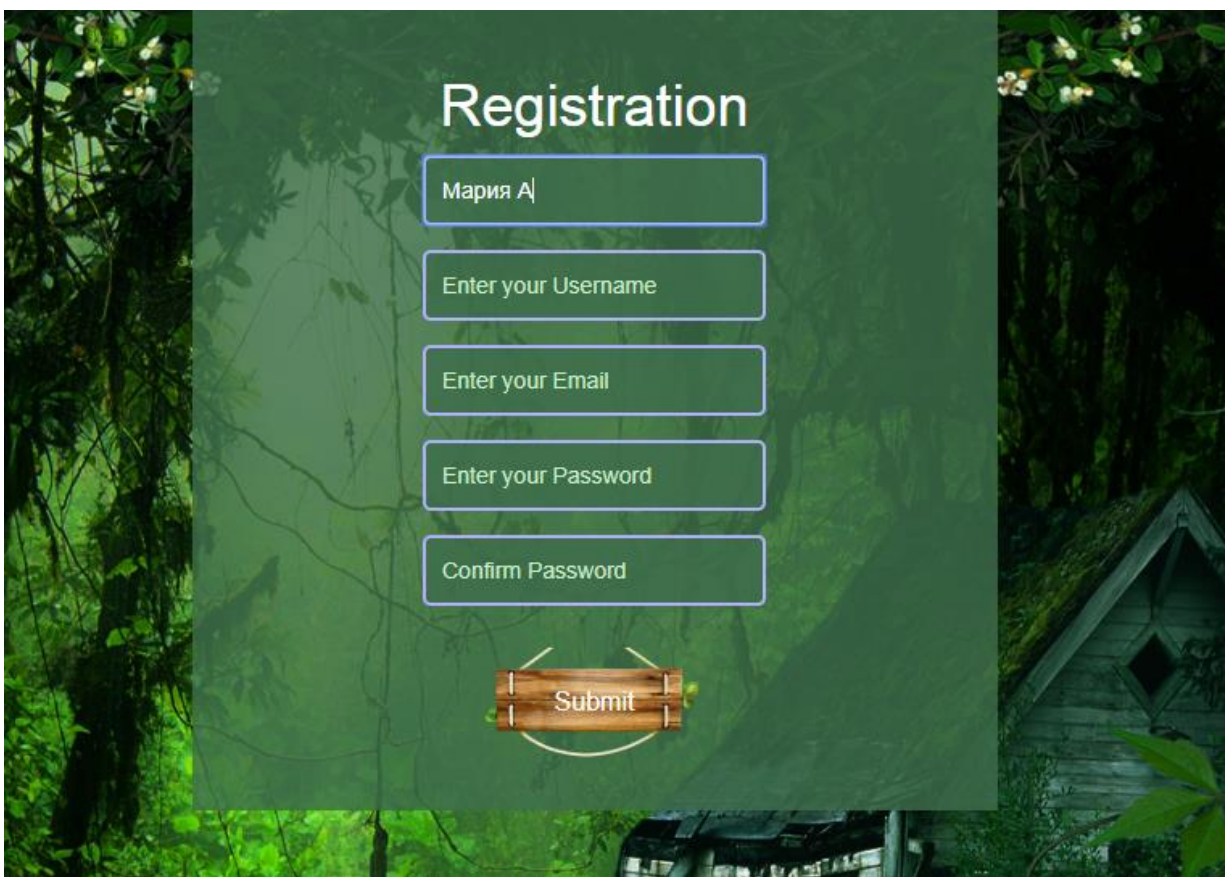


фигура 14 Facebook popup window

### 3.3. Регистрация

Ако все още потребителят не е регистриран, натискайки бутона "Registration", той се изпраща към формата за регистрация. В предложената форма следва да се въведат личните имената на потребителя, потребителско име, имейл и парола. Информацията се изпраща към базата данни, чрез натискане на бутона "Submit".



A registration form is centered on a semi-transparent green rectangular background. The background image shows a dense forest with green foliage and a wooden cabin roof visible on the right. The form has a title 'Registration' at the top. Below it are five input fields: the first contains 'Мария А|', the others are empty and labeled 'Enter your Username', 'Enter your Email', 'Enter your Password', and 'Confirm Password'. At the bottom is a 'Submit' button designed to look like a wooden log with metal bands and a small green handle on the right.

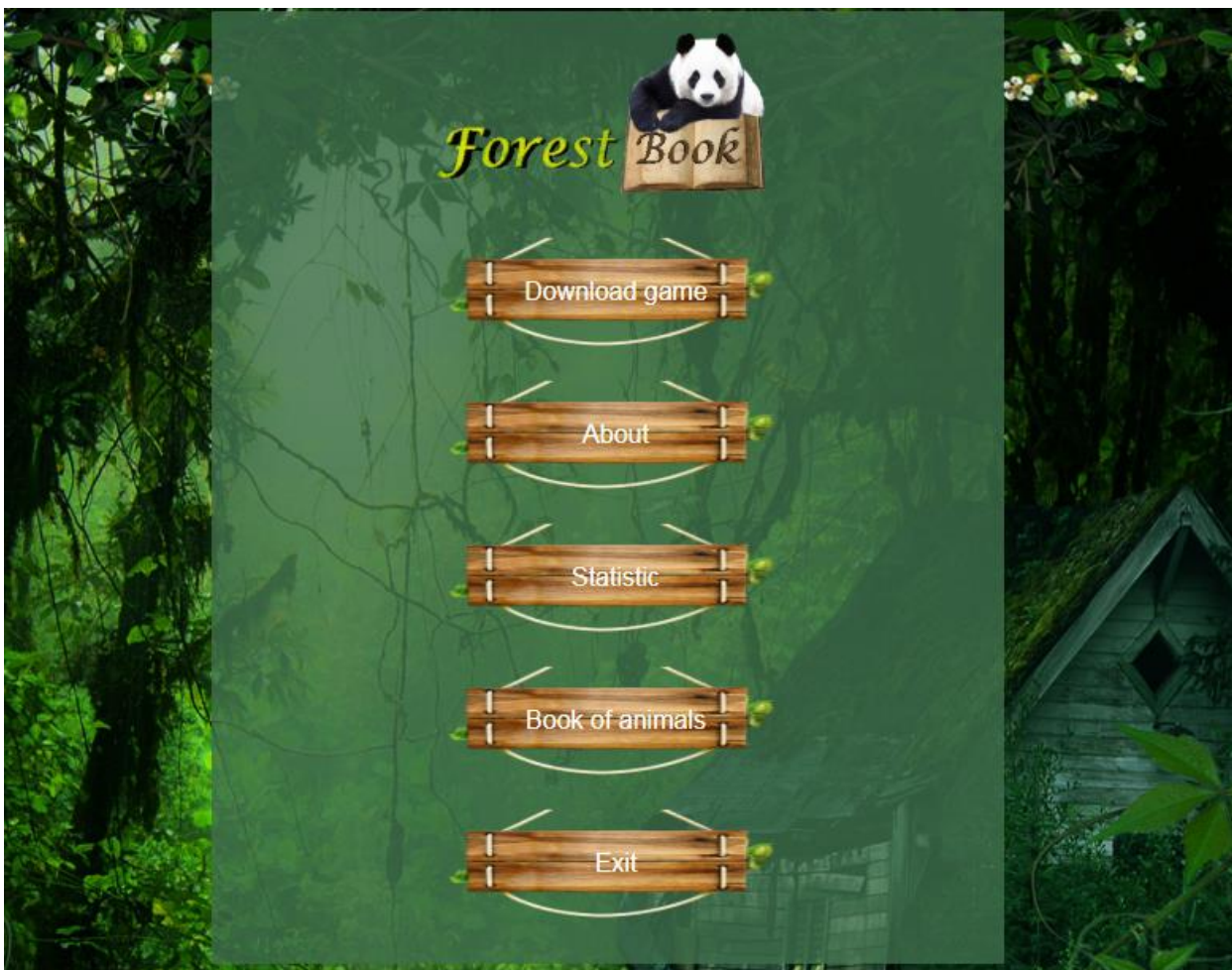
# Registration

фигура 15 Форма за регистрация

С успешното аутентикиране на потребителя, той се допуска до основното меню, което предлага възможност да се изтегли играта и да се проследи напредването в нея.

### 3.4. Меню на сайта

Менюто предоставя следния набор от бутони: "Download game", "About", "Statistic", "Book of animals" и "Exit".

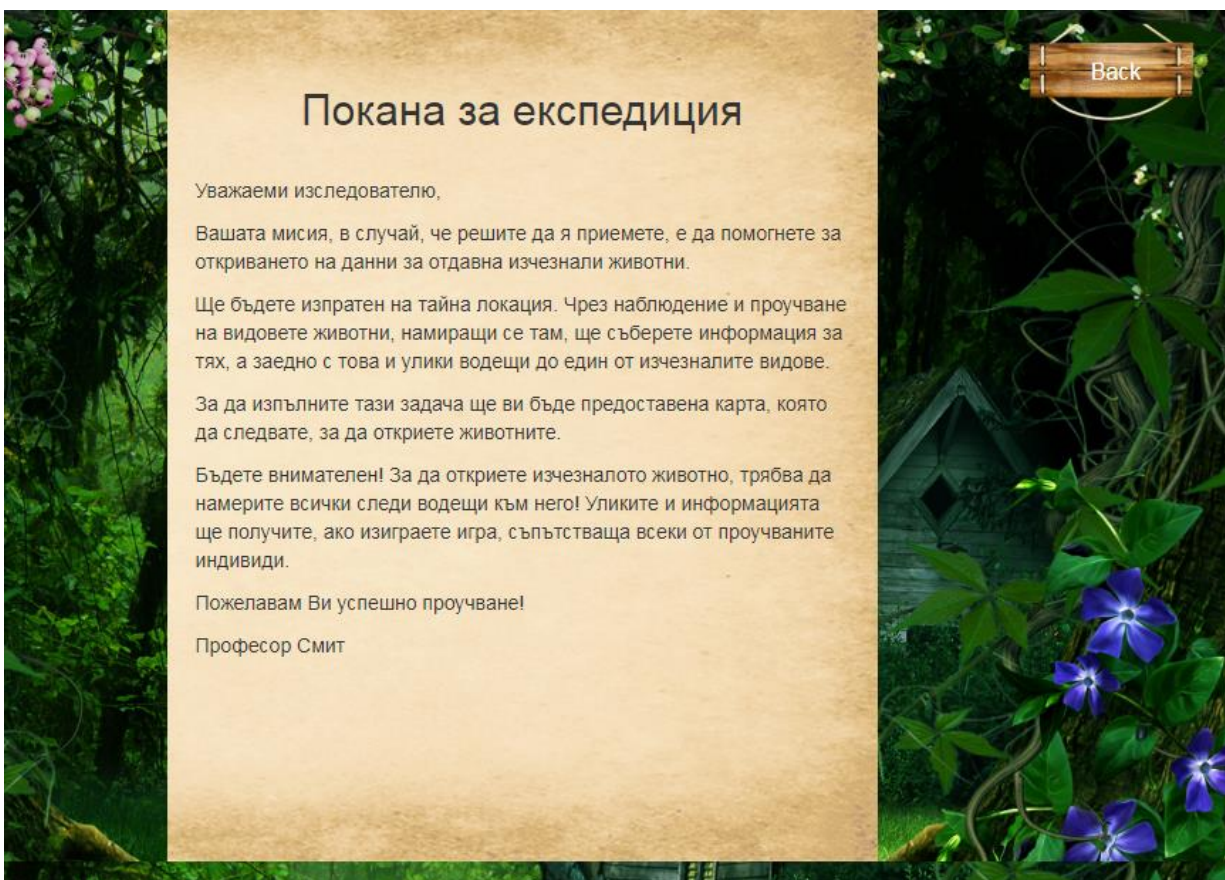


фигура 16 основна страница на сайта - меню

- Чрез бутона "Download game", потребителя може да изтегли играта на своето десктоп устройство.
- Натискайки бутона "About", пред потребителя се показва кратко описание на правилата на играта поднесено под формата на писмо, с цел събуждане на интереса и привличане на вниманието.
- Бутонa "Statistic" визуализира таблица с резултатите на всички играчи.
- Бутонa "Book of animals" отвежда потребителя към flip book книгата
- Бутонa "Exit" позволява на потребителя да излезе от акаунта си и се препраща на началната страница.

### 3.5. Относно играта (About)

Това е една кратка опознавателна страница, съобщаваща основните линии на развитие на играта. Страницата е придружена с бутон връщащ към основното меню.



фигура 17 страница About

### 3.6. Статистика

Тази страница показва таблица с резултатите, извикани директно от базата с данни „Firebase“ за всички потребители, които са направили регистрация, тяхното място в ранглиста – на кое ниво са и колко точки от решените пъзели са натрупали, като за връщане в основното меню може да се използва бутон назад „Back“. Статистиката може да се проследи и от менюто на самата игра.



Statistic		
Player	Level	Score
Aleksandar Petrov	20	60
Димитър Цветков	18	50
Мария Атанасова	15	36
Иван Иванов	13	20

фигура 18 Таблица със статистика за играчите

### 3.7. “Book of animals”

Бутона “Book of animals” препраща към HTML страница, на която под формата на flipbook се визуализира цялата събрана до момента от играча информация за животните.



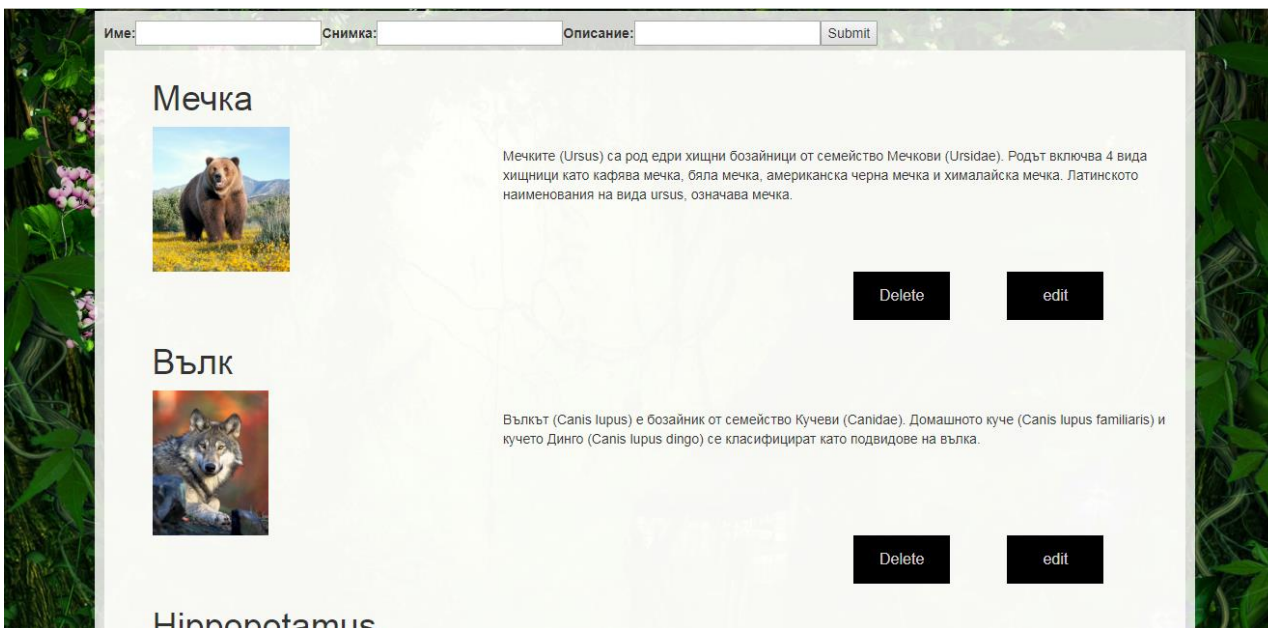
фигура 19 Book of animals - flip book

### 3.8. Административен панел

След успешна проверка на административните права, администратора се допуска до показаната страница.

Предоставя му се възможност за въвеждане на нови животни в полетата на Firebase. Следва да се добави възможност за промяна на съдържанието за вече съществуващите в базата данни животни и възможност за премахването им, като към момента тези функционалности не са готови. Видно от фигурата по – долу, за тях са предвидени бутони.

Всички животни от базата данни се визуализират на страницата след въвеждането им.



фигура 20 Административен панел

## Заклучение

Тази дипломна работа представлява разработка на уеб сайт, кореспондиращ със създадените от другите членове на проекта база данни и Unity игра. Целта на система е обогатяване на знанията на потребителите относно животинския свят, като в същото време да се забавляват, играейки.

За разработката се използва ГРС. Описани са първите два етапа на разработка на проекта . След края на втория етап основните функционалности са изградени успешно, което реализира основните задачи на дипломната работа.

Системата на сайта може да се доразвие и обогати, чрез изпълнение на следните задачи:

- Създаване на профилни страници на потребителите
- Въвеждане на допълнителни мини тестчета, за проверка на натрупаните знания, решаването на които да носи бонус точки на играча.
- Разширяване на възможностите на административния панел, чрез премахване и редактиране на информация за животните и потребителите.

Това ще допринесе за цялостното завършване на функционалностите на сайта.

## Библиография

---

<sup>1</sup> <https://stackshare.io/posts/the-react-story>

<sup>2</sup> <https://medium.jonasbandi.net/angular-vs-react-popularity-ea2659308cd5>

<sup>3</sup> <https://reactjs.org/>

<sup>4</sup> [https://bg.wikipedia.org/wiki/AngularJS#cite\\_ref-angularishere\\_1-0](https://bg.wikipedia.org/wiki/AngularJS#cite_ref-angularishere_1-0)

<sup>5</sup> <https://docs.angularjs.org/guide/introduction>

<sup>6</sup> <https://insights.stackoverflow.com/survey/2017#technology-most-loved-dreaded-and-wanted-frameworks-libraries-and-other-technologies>

<sup>7</sup> <https://medium.com/@TechMagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d>

<sup>8</sup> <https://nodejs.org/en/about/>

<sup>9</sup> <http://www.turnjs.com/>

<sup>10</sup> <https://www.forbes.com/sites/johnkoetsier/2017/12/19/google-owns-100-of-2017s-top-10-fastest-growing-sdks-on-android/#4b04a1ea7598>

<sup>11</sup> <https://adtmag.com/articles/2017/06/23/android-sdk-report.aspx>

<sup>12</sup> <https://instabug.com/blog/game-engines/>