



НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
Магистърски Факултет
Департамент Информатика
Програма „Софтуерни технологии в Интернет”
Специализация „Проектиране и разработване”

МАГИСТЪРСКА ТЕЗА
на тема:

Онлайн игра „Бинго“

Дипломант:
Михаил Габеров, Ф. №: 58116

Научен ръководител:
доц. д-р Венцислав Джамбазов

София, 2020г.

Съдържание

Онлайн игра „Бинго“	1
Съдържание	2
Увод	3
I. Преглед на предметната дейност	3
II. Анализ на функционалността на приложения с подобна предметна област	5
III. Използвани технологии	6
Изложение	8
IV. Проектиране	8
A. Анализ на заданието и описание на изискванията	8
B. Логически дизайн	10
V. Общ преглед:	10
A. Архитектура на приложението – подсистеми, модули, и връзки между тях:	10
B. Конфигурация - конфигурациите в приложението могат основно да се разделят на няколко вида:	17
C. Настройки/технически зависимости за фронт-енд частта на приложението	19
D. Настройки/технически зависимости за бек-енд частта на приложението (включващи админ и сървър частите)	22
E. Графично представяне на структурата на приложението	29
VI. Технически характеристики	32
A. Обем на приложението – за определяне на обема на приложението като брой файлове и брой на редове код е използван инструментът CLOC (count lines of code):	32
B. Responsiveness – “отзовчив” дизайн	35
C. Конфигурируеми настройки - както беше споменато по-рано, всички предварителни конфигурации на играта се намират във файла config.json. Те са както следва:	43
D. Изводи за техническата стойност на приложението:	44
VII. Взаимовръзки на отделните подсистеми	48
A. Клиентска част	48
B. Back-office панел	50
C. API	54
D. Discoverer	55
VIII. Реализация	59
A. Имплементация	59
B. Валидация на приложението	64
C. Ръководство за инсталiranе	66
D. Ръководство за потребителя	68
IX. Възможности за промяна	73
A. Разширяване възможностите на административната част	73
B. Разширяване функционалностите на самата игра:	75

Заключение.....	76
X. Използвана литература.....	77
XI. Приложения.....	78
A. Каталог на схема дефинициите на колекциите в MongoDB.....	78
B. Каталог на използваните файлове	85

Увод

I. Преглед на предметната дейност

Бингото е една от най-разпространените игри в света. За него не са необходими нито познания, нито следване на стратегии за игра. Всичко необходимо, за да спечелите на **бинго**, е да имате късмет. Затова много хора го предпочитат като начин за разпускане и евентуална печалба.

Онлайн бингото се играе в интернет зала в съответствие с правилата на класическото бинго. Всеки играч купува една или няколко карти с комбинация от числа. След започване на бинго играта се теглят числа с помощта на ГСЧ -

генератор за случайни числа. Изтеглените числа се маркират ръчно от играча или автоматично в бинго картата, докато на някоя от закупените карти от играчите не излезнат всички числа и спечели "бинго".

Една от най-популярните бинго игри е Бинго със 75 топки, каквато е реализирана в текущата дипломна работа.

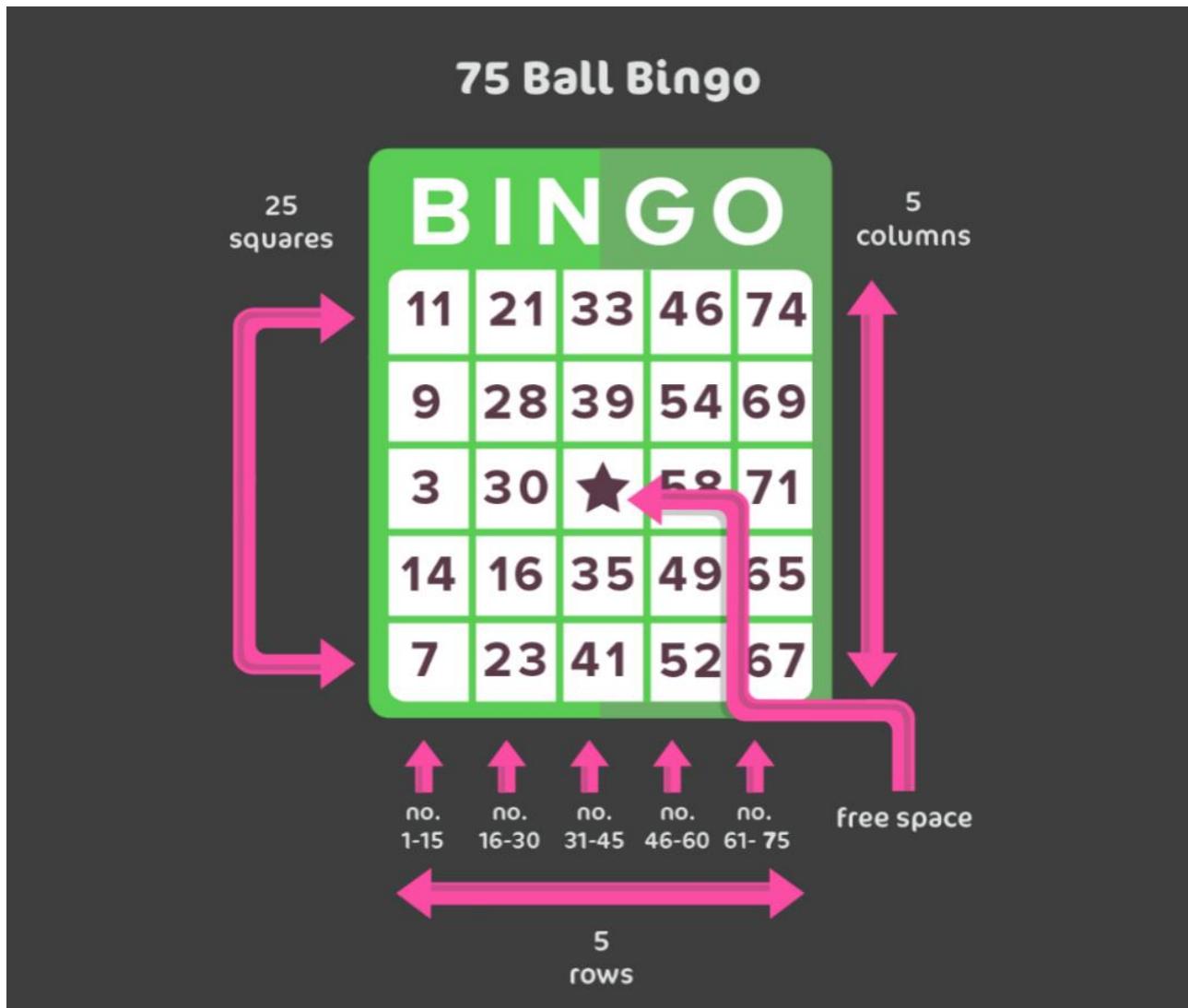


Figure 1: Бинго игра с 75 топки

Бинго със 75 топки е най-популярната игра в Северна Америка. Бинго числата се асоциират с буквите в думата БИНГО. Разновидности: Класическо - 25 числа на талон 5x5; Свободно пространство - 24 числа на талон 5x5 със свободно място в центъра; Daub-All - 75 или 72 числа на 5x5 полета.

В настоящата дипломна работа е реализиран ваиранта „Свободно пространство“ – 24 числа на талон 5x5 със свободно място в центъра.

Цел на играта:

- Играчите избират и закупуват бинго карти.

- Бинго числата (топките) се изтеглят една по една.
- Награда се дава на всеки играч, който запълни всички числа принадлежащи към даден модел за победа – хоризонтална линия, вертикална линия, диагонал или четирите крайни числа на една бинго карта.
- Играта приключва, когато бъдат изтеглени определен брой топки – в нашия случай това е конфигурируема величина.

II. Анализ на функционалността на приложения с подобна предметна област

Бинго е хазартна игра от категорията на числови лотарийни игри. Предлагането на хазартни онлайн игри в световен мащаб е огромно, като в частност онлайн бингото завзема своята немалка част. Онлайн класическото бинго, точно като своят традиционен колега, е игра на късмет и шанс, който изисква минимални умения от играчите.

Темата на текущата дипломна работа е бинго игра със 75 топки и затова разглежданите варианти за съществуващи приложения са от този тип.

Онлайн бинго със 75 топки е американската версия на бинго. Ето защо тази версия на онлайн бинго се различава от бинго игра с 90 топки по няколко причини. На първо място, броят на топките е по-малък (75 вместо 90). Второ, класификацията на печелившите комбинации е по-различна. Трето, видът на картата бинго е различен. Всяка бинго карта има 5 реда и 5 колони с 24 произволно избрани числа и едно „празно“ място в средата, което се смята за свободно пространство и не участва в самата игрова логика. Освен това колоните са означени с букви, които се комбинират заедно, за да образуват думата BINGO (т.е. първата колона е обозначена с буквата B , втората с буквата I и т. н.).

Изследователите предполагат, че една от първите бинго игри, появили се онлайн, е била [Bingo Zone](#), следвана от [Bingo Blitz](#) в средата на 90-те години. Днес онлайн бинго индустрията е също толкова успешна и продължава да се подобрява непрекъснато. Софтуерните компании харчат милиони, за да се разработват по-реалистични и по-удовлетворяващи онлайн забавления за всички фенове на Бинго. Така че в близко бъдеще бинго феновете могат да очакват още по-интересни и вълнуващи онлайн бинго игри.



Figure 2: Online Bingo

Бесплатни бинго онлайн игри се предлагат в множество сайтове. Въпреки това, само някои от тях ви предлагат абсолютно бесплатно бинго. Една от тези бесплатни възможности ще бъде първоначалната версия на текущата разработка.

III. Използвани технологии

Използваните технологии в текущата разработка можем условно да разделим на няколко вида:

- Front-end – технологии използвани при разработката на потребителските интерфейси и функционалност предназначени за крайния потребител (играч).
- Back-end – технологии използвани при разработката на „скритите“ функционалности за крайния потребител, т.е. програмната логика осигуряваща връзка на фронт-енд частта със сървъра и базата данни.
- Testing – технологии използвани за имплементиране и изпълнение на „юнит“ тестове (unit tests), които служат за подсигуряване правилното функциониране на съществуващите модули.
- Database – технологии използвани за имплементиране на база данни нужна за съхранението и обработката на данните на регистрираните

играчи и съответно служебните лица, т.нар. администратори, които имат пълномощия за манипулиране на техните данни чрез създадено за целта back-office приложение.

Технически спецификации: [SPA \(single-page application\)](#) приложение/игра предназначено за хазартната онлайн индустрия. Използваните технологии в са както следва:

- **Клиентска част:** [HTML5](#), [CSS \(SASS\)](#), [ES6](#) и [ReactJS](#) използван за административната част



- **Тествове (unit tests):** [Mocha and Chai](#), [React Testing Library](#)



- Сървърна част: [Express/Node.js](#)



- База данни: [MongoDB](#)



Изложение

IV. Проектиране

A. Анализ на заданието и описание на изискванията

Описание: Проектът ще представлява стандартна „Бинго“ игра със 75 топки – американски тип, която ще може да се играе във браузър. Проектът ще бъде SPA приложение ([single page application](#)) като при разработката му ще бъде спазван тест ориентиран подход - [Test-Driven Development \(TDD\)](#). Главните функционалности и игровата логика ще бъдат подсигурени с юнит тестове ([unit tests](#)). Игровата последователност ([user journey](#)) ще бъде както следва:

1. При първоначално отваряне на приложението – форма за регистрация или форма за вход (login) за вече регистрирани потребители, притежаващи необходимите валидации на потребителските данни (имейл, парола).

2. След успешна регистрация или логин – първоначален игрови еcran показващ основните игрови компоненти на играта: профил с потребителските данни, баланс с моментно разполагаемата виртуална валута и контроли предлагани опции за закупуване на бинго карти.
3. След закупуване на желан брой бинго карти (от 1 до 8 възможни) и натискане на бутона „Старт“ - втори игрови еcran показващ прогреса на самата игра – закупените бинго карти, върху които играта може да маркира изтеглените числа, „тръба“ показваща с анимация последните пет изтеглени топки и сфера симулираща „разбъркването“ на топките с анимация наподобяваща този ефект. Предвижда се също така модул показващ, с кратка анимация, възможните печеливши модели – хоризонтална линия, вертикална линия, диагонали или четирите краища на бинго картата.
4. След приключване на дадена игра, т.е. след изтегляне на предварително зададен, чрез конфигурация, брой топки – всички компоненти спират „действието“ си, което ще бъде илюстрирано чрез стопиране на анимациите. На играта ще бъде показан за кратко време „рорир“ еcran с резултата и ОК бутон, чрез който ще бъде препратен отново към екрана за покупка на карти.
5. При натискане на бутона ‘Logout’ всички игрови елементи ще бъдат премахнати от екрана и на играта ще бъде предоставена възможност за нова регистрация или вход (login) чрез съответните форми.
6. Системата ще разполага с админ панел (back office) за управление на потребители, който ще предлага следните възможности:
 - a. Форма за вход (login) на потребител с роля администратор
 - b. След успешен логин, на администратора ще бъде показан еcran с контроли, с които:
 1. Ще може да чете информацията за регистрираните потребители
 2. Ще може да добавя нови потребители
 3. Ще може да редактира информация за текущо регистрирани потребители:
 - Промяна на име
 - Промяна на парола
 - Промяна на сумата по баланса на играта, т.е. да „раздава“ награди във вид на виртуална валута, с която играта ще може да закупува карти за игра или да отнема такава като „наказание“
 - Промяна броя победи – тоест броя спечелени бингота
 4. Ще може да изтрива съществуващи потребители

B. Логически дизайн

1. Архитектура на приложението

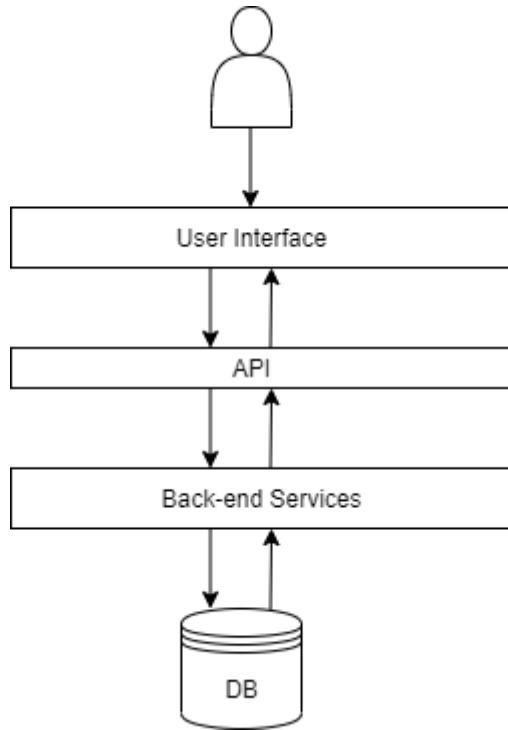


Figure 3: Архитектура на приложението

V. Общ преглед:

A. Архитектура на приложението – подсистеми, модули, и връзки между тях:

- Front-end функционалност - това, което вижда крайния потребител и с което се изпълнява една игра, е разпределена в следните директории, като за начална точка (entry point) или стартовата логика на приложението се счита логиката имплементирана във файла [app.js](#) (обяснена в края на тази точка):
 - a. /blower – съдържа единствен файл [blower.js](#), този файл се състои от класовете **Ball** и **Blower** отговарящи за логиката имплементираща анимацията на топките, показващи номерата от 1 до 75 в т. нар. „туба“, или сферовиден обект, в който се движат топките с произволна скорост и посока и разполагащ с отвор в горната част, от където излизат изтеглените топки/номера.
 - b. /card - съдържа следните файлове:
 - i. [card.js](#) – съдържа клас **Card**, отговаря за създаването на обект „карта“ в DOM

- (document object model) модела и имплементира логиката, отговаряща за избиране (кликане) на даден номер от карта, определянето му на това дали вече е изтеглен, както и на това дали е спечелено „бинго“, тоест дали е изпълнен някой от зададените печеливши модели (winning patterns).
- ii. [card-drawer.js](#) – съдържа клас ***CardDrawer***, отговаря за изрисуването на генерираните карти (използвайки обекти от класа Card), както и за тяхното визуализиране или скриване от екрана при начало или край на играта.
 - iii. [card-generator.js](#) – използва клас ***CardNumbersGenerator***, за да генерира зададен брой карти, съдържащи случайно генериирани числа в тях.
 - iv. [card-numbers-generator.js](#) – генерира Бинго карта от американски тип с 5 колони, съдържащи съответно случайно генериирани числа за колона, т.е. от 1 до 15 в първата колона, от 15 до 30 във втората, от 30 до 45 в третата, от 45 до 60 в четвъртата и от 60 до 75 в петата последна колона.
- c. /dauber – съдържа следните файлове:
- i. [ball.js](#) – отговаря за създаването на обект тип Ball и анимирането ѝ до достигане на крайна позиция. Използва клас ***Animator*** за извършване на анимациите и обект от тип ***PubSub***, който приема като входен параметър при създаването на обекта (в конструктора си), реализиращ Publish/Subscribe механизъм служещ за управление на събития. В случая, обект от тип ***Ball*** изпраща събитие за изтеглена пета топка (FIFTH BALL DRAWN), което уведомява т.нап. ***dauber*** обект да актуализира текущата анимация на изтегляните топки. Също така тук се намира функционалността определяща различните действия анимиращ топките съответно при първата фаза – вертикално излизане от сферата и втората –

хоризонталното им преместване на ляво по „тръбата“. Анимациите могат да използват различни методи за работа, които се определят в класа *Animator*, основаващи се на различни математически функции от типа на линейна, квадратна и т.н.

- ii. [dauber.js](#) – съдържа класа *Dauber*, отговарящ за създаването и управлението на т. нар. обект „даубер“, или мястото в игровата логика където се „слуша“ за начало и край на играта и се приедприемат съответните действия. При начало на играта се стартира тегленето на топките, случайни числа, подчиняващи се на правилата на играта „Бинго“ от американски тип. Тук се определя интервала (време в секунди), през който се теглят топките и който по подразбиране е зададен в конфигурационния файл [config.json](#). Също така този клас имплементира логиката отговаряща за тегленето на нова топка (число), за анимирането на видимите вече топки и за край на играта, т.е. инициирането на събитие, което се прихваща от интересуващите се компоненти с цел обновяване на тяхното състояние. Тук е важно да се спомене, че по-голяма част от реализираните събития в играта се реализират чрез използване на основна функционалност в JavaScript, а именно тази предоставена от *CustomEvent* класа и съответно методите *dispatchEvent* и *addEventListener*, които се прилагат към DOM елементи и са достъпни в глобалния контекст (scope). Това е направено така с цел да се избегне използването на допълнителни външни библиотеки, реализиращи подобен тип поведение (като jQuery, Backbone и т.н.), както и с образователна цел, а именно по-задълбочено използване на чист JavaScript код, макар това да води до по-голямо количество „ръчен“ (boilerplate) код.

- d. /events – съдържа следните файлове:

- i. [events-consts.js](#) – съдържа константен обект с използваните събития във фронт-енд логиката на играта, чиито имена са зададени като константи, с цел преизползването им навсякъде където е необходимо. Подобен подход носи предимството на преизползваемост на кода вместо т.нар. „hard coding“ на самите имена във вид на string обекти. Тоест, вместо директното изписване на името на събитието като стринг, там където е необходимо неговото използване се „вмъква“ въпросния клас и така за име на събитието се използва константната му величина.
 - ii. [pubsub-service.js](#) – този клас съдържа пристапа реализация на Publish/Subscribe интерфейс, реализиран от David Walsh (<https://davidwalsh.name/pubsub-javascript>). Използва се на няколко места в играта, като основна подбуда за въвеждането му е образователната цел или още един начин за реализиране на асинхронно програмиране. Основната му идея е, че класовете, които го използват могат да използват предоставените от него методи за, съответно, публикуване на събития от даден тип и тяхното прихващане от интересуващите се страни, т.е. от други класове, чиято логика се основава на дадено събитие. Т.нар. абонати или *subscribers*. За пример виж класовете *ApiController* и *ViewManipulator*.
- /local-storage – съдържа следните файлове:
 - a. [local-storage-service.js](#) – съдържа класа *LocalStorageService* реализиращ интерфейс за работа с обекта localStorage имплементиран в съвременните браузъри като част от HTML5 спецификацията. LocalStorage е ограничена памет в клиентската част, в случая браузъра, където могат да се запазва определена информация използвана от приложението. Наподобява т.нар. „бисквитки“, които бяха по-популярни в миналото като метод за запазване на информация

на клиентската машина. Много често се използва за запаметяване на данните от потребителски вход с идеята да се предотврати повторно искане към потребител, който вече е легитимиран. В нашия случай *LocalStorageService* предоставя методи за четене и запис на логин токен за играчи, баланс на играча, логин токен за администратори, както и методи за проверка на вече авторизиран играч. Също така разполага с метод, който се извиква при излизане (logout) на играча от играта, при което данните записани в local storage се изтриват, което предотвратява непозволени действия за неавторизирани потребители.

- b. /market-place – съдържа следните файлове:
 - i. [market-card.js](#) – съдържа логиката, която отговаря за опциите за закупуване на карти, които играча вижда, когато влезе в играта със своето потребителско име и парола. Този клас разполага с методи за определянето на броя закупени карти, за действителното им закупуване, т.е. изпращане на заявка към базата данни чрез ApiController класа, както и за определяне на цената за бинго карта, чиято стойност е зададена по подразбиране в конфигурационния файл config.json. Тук се намира и логиката, която се изпълнява когато играча няма достатъчно средства за закупуването на избрания брой карти. В този случай приложението показва pop-up съобщение на играча, съобщавайки му текущия баланс и подканвайки го да направи депозит.
 - c. /utils – съдържа следните файлове, предоставящи преизползваема функционалност, т.е. такава, която да не бъде специфична за даден компонент или модул, а за приложението като цяло:
 - i. [animator.js](#) – съдържа клас *Animator*, предоставящ методи за изпълнение на различен тип анимации.
 - ii. [bangup.js](#) – съдържа клас *Bangup*, предоставящ функционалност за постигане на т. нар. „бангъп“ ефект, т.е. анимирано

- визуализиране на натрупването на дадена сума от начало X до край Y.
- iii. [numbers-generator.js](#) – съдържа клас ***NumbersGenerator***, предоставящ функционалност за случайно генериране на числата от една бинго карта по колони, т.е. първа колона от 1 – 15, втора от 15 – 30 и т.н.
 - iv. [timer.js](#) – съдържа клас ***Timer***, отговарящ за създаването обект тип „брояч“, тоест секундите, които се отброяват например преди старт на нова игра. Разполага с метод за стартиране на дадено събитие при край на отброяването.
 - v. [utils.js](#) - предоставя константен обект с общо преизползвани методи, чиято идея е да бъдат използвани където е необходимо в приложението, например метод за елиминиране на дубликатите в даден масив – *eliminateDuplicates(arr)*.
 - vi. [view-manipulator.js](#) – предлага статични методи, служещи за манипулиране на състоянието на различни компоненти в приложението, като например съобщения за грешка, информация за играла (показана в горния десен ъгъл на екрана), обновяването на баланс сумата на играла, функционалност за показване или скриване на даден DOM елемент. При инициализация на този клас се селектират основни елементи в приложението (форма за регистрация, форма за логин, карти за покупка и т.н.). Също така този клас отговаря за „закачането“ на необходими „слушатели“ (listeners) за край на играта, с цел обновяването на състоянието на засегнатите игрови компоненти. Използва *subscribe* метода, предоставен от ***PubSub*** модела, за реализиране на част от функционалността отговаряща за разпространение и прихващане на събития в приложението.
 - vii. [winning-patterns.js](#) – този клас съдържа логика отговаряща за определянето на

печелившите „Бинго“ модели, т.е. комбинациите от числа, носещи победа на играта. Печелившите модели са 4 на брой, представени от 4-те метода в класа:

1. HorizontalPattern – изтегляне на 5 числа по хоризонталата – хоризонтална линия.
 2. VerticalPattern – изтегляне на 5 числа по вертикалата – вертикална линия.
 3. DiagonalPattern – изтегляне на 5 числа по един от двата диагонала - ляв или десен.
 4. CornersPatterns – изтегляне на 4 числа отговарящи на четирите ъглови стойности зададени в бинго картата.
- d. /winning - съдържа следните файлове, предоставящи функционалност, реализираща общи елементи използвани при край на играта с печеливш резултат или по време на нейното провеждане:
- i. win-patterns-anim-module.js - имплементира анимиран компонент, чиято цел е да показва печелившите модели на играта по време на самата игра с кратка анимация.
 - ii. flying-prize.js – имплементира клас, чиято функция е да анимира спечелена награда (цифра), премествайки я с кратка анимация от средата на екрана до горния десен ъгъл където се намира баланс сумата на играта . След края на тази анимация се задейства bangup функционалността, която показва забързано изброяване на новата баланс сума.
 - iii. winning-dialog.js – имплементира логика, отговаряща за визуализирането на диалогов прозорец, показващ се, при край на играта, който съдържа информация за крайния резултат от играта – загуба, т.е. никој едно спечелено бинго, спечелени едно, две или повече, със съответните графични елементи (намръщено, усмихнато или засмяно човече в горната част на диалоговия прозорец,

- както и кратки анимации, обозначаващи броя спечелени бинга).
- e. /initializer – съдържа единствен файл с клас *Initializer* със статични методи отговарящи за първоначалното стартиране на играта. Задава необходимите начални настройки на игровите компоненти и ги визуализира на екрана.
 - f. app.js – входната точка към играта. В този файл е съсредоточена логиката, обединяваща всички игрови модули и тяхното взаимодействие. Тук се зареждат първоначалните данни от конфигурационния файл (config.json) и се имплементира тяхната употреба чрез класа *Initializer*. Например метаданните, които се използват за приложеното, конфигурационни променливи, определящи времето за теглене на топка, броя топки определящи до кога продължава една игра и т.н.

B. Конфигурация - конфигурациите в приложението могат основно да се разделят на няколко вида:

1. Игрови настройки – config.json файл съдържащ следните опции, във вид на json обект наречен *gameConf*:
 - i. id – идентификационен номер на играта (в конкретния случай без значение, за пример е зададен като единица, но вида му може да варира).
 - ii. name – име на играта, в случая е зададено като „American Bingo”, какъвто е реализирания тип игра.
 - iii. numbers – масив, съдържащ 75-те числа, отговарящи на изискванията за една „Бинго“ игра от американски тип.
 - iv. skin – обект *skin* с единствено свойство *name*, което определя използваната „кожа“ (skin) в играта. Идеята за в бъдещи версии на играта е да може да се използва с различни „кожи“ (skins), което да бъде реализирано само чрез промяна на тази конфигурационна променлива. Подобна промяна би била например промяна на основните игрови цветове, графични елементи и т.н. В бъдеще това би било означавало и различни звуци и/или

- анимации определяни от избраната „која“ (skin).
- v. appTitle – стринг съдържащ мета заглавието на играта или това, което се визуализира в таба на браузъра (напр. *Welcome To Bingo Bigul*) и което се изписва в HTML таг `<title>`.
 - vi. turnsCount – брой ходове в една игра, след изтичането им играта приключва.
 - vii. freeSpotImgPath – стринг, съдържащ пътя до графично изображение, което се използва при визуализирането на бинго карта. Показва се в средата на картата, в т. нар. бесплатна зона или *free spot*.
 - viii. drawIntervalSeconds – числов променлива, съдържаща времеви интервал в секунди, който се използва при теглене на нова топка. Тоест всяка следваща топка бива изтеглена през интервала от време зададен чрез тази променлива. По подразбиране в нормална бинго игра този интервал е 6-7 секунди, но може да бъде променян когато се реализират различни игрови режими (забързване, забавяне и т.н.). В бъдещи версии на играта се предвижда функционалност, която ще позволи на играта да паузира тегленето на нови числа/топки или настройването на времевия интервал, чрез предоставени на играта настройки от меню.
 - ix. beforeStartGameSeconds – числов променлива съдържаща брой секунди, които се отброяват от бояча преди начало на играта.
 - x. marketCards – флаг, чрез който се определя дали елемента *карти за покупка (market cards)* е включен/видим в приложението.
 - xi. dauber – флаг, чрез който се определя дали елемента *даубер (dauber)* е включен/видим в приложението.
 - xii. playingCards – флаг, чрез който се определя дали елемента *карти за игра (playing cards)* е включен/видим в приложението.

- xiii. mainGame – флаг, чрез който се определя дали елемента ***основна игра (main game)***, т.е. възможността за стартиране на игра, е включен/видим в приложението.
- xiv. winningDialog – флаг, чрез който се определя дали елемента ***диалогов прозорец за победители (winning dialog)*** е включен/видим в приложението.
- xv. cardPrice – числова променлива, определяща цената във вид на виртуална валута за една бинго карта.
- xvi. winPatternsAnimModule – флаг, чрез който се определя дали елемента, показващ анимация с печелившите модели е включен/видим в приложението.

C. Настойки/технически зависимости за фронт-енд частта на приложението

1. За правилното настройване и стартиране на фронт-енд частта на приложението се използва [node package manager – npm](#). Софтуерните пакети, които се инсталират с негова помощ са разделени на две групи – **dependencies** и **devDependencies** (от **development**), разделението е продиктувано от начина на употреба на **npm**. Самите пакети са:

```
"dependencies": {
  "body-parser": "^1.18.2",
  "dotenv": "^2.0.0",
  "express-jwt": "^5.3.1",
  "jsonwebtoken": "^8.5.1",
  "node-fetch": "^2.6.0",
  "passport": "^0.3.2",
  "passport-local": "^1.0.0"
}

"devDependencies": {
  "@babel/core": "^7.1.6",
  "@babel/plugin-transform-runtime": "^7.1.0",
  "@babel/polyfill": "^7.0.0",
  "@babel/preset-env": "^7.7.6",
  "@babel/preset-flow": "^7.0.0",
  "@babel/preset-react": "^7.0.0",
  "@babel/register": "^7.0.0",
  "@babel/runtime": "^7.1.5",
  "@testing-library/react": "^9.4.0",
  "autoprefixer": "^6.7.7",
  "babel-core": "^6.26.3",
  "babel-eslint": "^9.0.0",
  "babel-jest": "^23.6.0",
  "babelify": "^10.0.0",
  "bootstrap-sass": "^3.3.7",
}
```

```
"browserify": "^16.2.3",
"classnames": "^2.2.5",
"del": "^2.2.2",
"event-emitter-es6": "^1.1.5",
"express": "^4.16.3",
"fbemitter": "^2.1.1",
"gulp": "^4.0.2",
"gulp-babel": "^8.0.0",
"gulp-concat": "^2.6.0",
"gulp-eslint": "^6.0.0",
"gulp-istanbul": "^1.1.3",
"gulp-jest": "^4.0.3",
"gulp-mocha": "^3.0.1",
"gulp-postcss": "^6.4.0",
"gulp-rename": "^1.2.2",
"gulp-sass": "^4.0.2",
"gulp-server-livereload": "^1.8.4",
"gulp-sourcemaps": "^1.12.1",
"immutable": "^3.8.2",
"invariant": "^2.2.4",
"jest-canvas-mock": "^2.2.0",
"jest-cli": "^24.9.0",
"jest-fetch-mock": "^3.0.0",
"mongoose": "^5.7.14",
"paper": "^0.12.4",
"react": "^16.12.0",
"react-dom": "^16.12.0",
"vanilla-modal": "^1.6.5",
"vinyl-buffer": "^1.0.1",
"vinyl-source-stream": "^2.0.0",
"watchify": "^3.11.1"
},
```

- a) за дефиниране и стартиране на различните части от приложението, като например стартиране само на фронт-енд частта или само на юнит тестовете за нея, се използва инструмент наречен [Gulp](#). Той се конфигурира посредством конфигурационен файл, намиращ се в главната директория на проекта – [gulpfile.js](#). В него се дефинират т.нар. задачи, които представляват групи инструкции, които да се изпълнят при стартиране на дадена задача. Самото програмиране на конфигурациите се имплементира с чист JavaScript. В нашия случай имаме дефинирани задачи за всеки от модулите изграждащи приложението. Така че всеки от тях да може да бъде стартиран самостоятелно по време на разработката и по този начин дава възможност на разработчикът по-бърза имплементация и по-лесно дебъгване в случай на грешка.
- b) Ето как изглежда част от конфигурационния файл на Gulp. Тук се виждат двете основни групи задачи, служещи за стартиране на фронт-енд и бек-енд частите на приложението,

както и експортираниятите команди, нужни за използването им чрез Gulp.

```
const build = gulp.series(clean, gulp.parallel(
  lint,
  scripts,
  scriptsBackOffice,
  scriptsDiscoverer,
  styles,
  stylesBackOffice,
  testsBackOffice,
  test,
  watch,
  webserver
));

const buildFrontEnd = gulp.series(clean, gulp.parallel(
  lint,
  scripts,
  styles,
  test,
  watch,
  webserver
));

const buildBackOffice = gulp.series(clean, gulp.parallel(
  scriptsBackOffice,
  stylesBackOffice,
  testsBackOffice,
  watch,
  webserver
));

exports.default = build;
exports.fe = buildFrontEnd;
exports.bo = buildBackOffice;
```

- c) Самото стартиране става чрез съответната команда в следния формат `gulp *команда*`. Например за да стартираме задачата, която отговаря за фронт-енд частта, използваме командата `gulp fe`, както се вижда на картиинката по-долу.

```
C:\Git\bingo (master -> origin) (bingo@1.0.1)
λ gulp fe
[14:03:08] Using gulpfile C:\Git\bingo\gulpfile.js
[14:03:08] Starting 'fe'...
[14:03:08] Starting 'clean'...
[14:03:08] Finished 'clean' after 67 ms
[14:03:08] Starting 'lint'...
[14:03:08] Starting 'scripts'...
[14:03:08] Starting 'styles'...
[14:03:08] Starting 'test'...
[14:03:08] Starting 'watch'...
[14:03:08] Starting 'webserver'...
[14:03:08] Finished 'lint' after 83 ms
[14:03:08] Finished 'scripts' after 99 ms
[14:03:08] Finished 'styles' after 104 ms
[14:03:08] Finished 'test' after 112 ms
[14:03:08] Finished 'watch' after 113 ms
[14:03:08] Finished 'webserver' after 125 ms
[14:03:08] Finished 'fe' after 210 ms
[14:03:09] Webserver started at http://localhost:8000
```

Figure 4: Стартиране на команда `gulp fe`

- d) Ако не подадем изрична команда, Gulp ще стартира задачата, която му е зададена по подразбиране. В нашия случай това е задачата *build*, чрез която се стартират всички подзадачи свързани.

```
exports.default = build;
```

D. Настойки/технически зависимости за бек-енд частта на приложението (включващи админ и сървър частите)

1. За база данни приложението използва небезизвестната, документно базирана, база данни [MongoDB](#) в комбинация с инструмента [Mongoose](#), който предлага ясно, схема базирано решение за моделиране да данни. Бек-енд функционалността е разпределена в следните директории:

- a) **routes** - за входна точка може да се счете файла [routes/index.js](#), където са дефинирани основните пътища (routes), към които се отправят http заявки за записване или четене на данни.
- b) **models** – съдържа следните файлове:
 - [admins.js](#): описва модела за базата данни за роля администратор, разполага със следните полета и методи:

- email – имейл адрес, с който е регистриран администратора
- name – име с което е регистриран администратора
- hash – служебно поле за запазване на хеширан стринг използван при създаване на парола
- salt - служебно поле за запазване на хеширан стринг използван при създаване на парола
- setPassword – метод за генериране на хеширана парола, използва полетата hash и salt
- validPassword – метод за валидиране на парола
- generateJwt – метод за генериране на уеб токен използван при регистриране на нови администратори или при логин на такива

- [users.js](#): описва модела за базата данни за роля играч, разполага със следните полета и методи:

- email – имейл адрес, с който е регистриран играча
- name – име, с което е регистриран играча
- hash – служебно поле за запазване на хеширан стринг използван при създаване на парола
- salt - служебно поле за запазване на хеширан стринг използван при създаване на парола
- balance –текущ баланс на играча във вид на виртуална валута
- wins – брой спечелени игри от играча
- setBalance – метод за обновяване на баланса

- setWins – метод за обновяване на броя спечелени игри
- setPassword – метод за генериране на хеширана парола, използва полетата hash и salt
- validPassword – метод за валидиране на парола
- generateJwt – метод за генериране на уеб токен използван при регистриране на нови играчи или при логин на такива

- [db.js](#): описва основните процеси за осъществяване на връзка с базата данни чрез mongoose

c) controllers – тук е описана програмната логика за работа с базата данни и моделите описани по-горе, т.е. това са т.нар. контролери. Тук може да се направи препратка към MVC (model-view-controller) модела, стандартен софтуерен дизайн шаблон, в който ролята на контролерите е да манипулира данните описани в модела и да ги презентира чрез презентационната логика описана във view частта. Съдържа следните файлове:

- [admins-auth.js](#): съдържа методи за регистрариране и вход на потребители с роля администратор
- [back-office.js](#): съдържа методи за извлечане, създаване, триене и обновяване на данни за играчите
- [profile.js](#): съдържа методи за управление на игровия профил на играча – четене на данни, обновяване на баланс и брой спечелени игри
- [user-auth.js](#): съдържа методи за регистриране и вход на играчи

d) config – съдържа единствен файл, [passport.js](#), използван от инструмента [Passport](#), в който са описани т.нар. стратегии за вход на роли играч и администратор

2. API – съдържа програмната логика за взаимодействие между фронт-енд и бек-енд частите на приложението, разпределена в следните файлове:

- a) [api-consts.js](#): съдържа всички пътища (routes) във вид на константни величини, чрез които се консумират услугите предоставяни от бек-енд частта, както и пътя за зареждане на конфигурационния файл (config.json) описан в предишната глава
- b) [api-controller.js](#): съдържа методи, които използват предложените от бек-енд частта услуги за манипулиране на играта, например вход и регистрация на играч, определяне на баланс или брой спечелени игри, и визуализирането на резултатите от тези действия, използвайки методите предложени от класа [ViewManipulator](#)
- c) [db-service.js](#): съдържа методи за работа с базата данни, възползва се от пътищата, описани по-горе като изпраща http заявки към тях, за изпълнение на съответно действие, като например вход на играч, нова регистрация или обновяване на текущия баланс на играча

1. Styles – изгледа на потребителските интерфейси на играта е реализиран чрез използването на [CSS \(cascading style sheets\)](#) и CSS препроцесора [SASS](#). Всички .scss (разширението идва от SASS) са разположени в директорията [styles](#). Тук се намират също така файловете, отговарящи за шрифта, използван в играта.

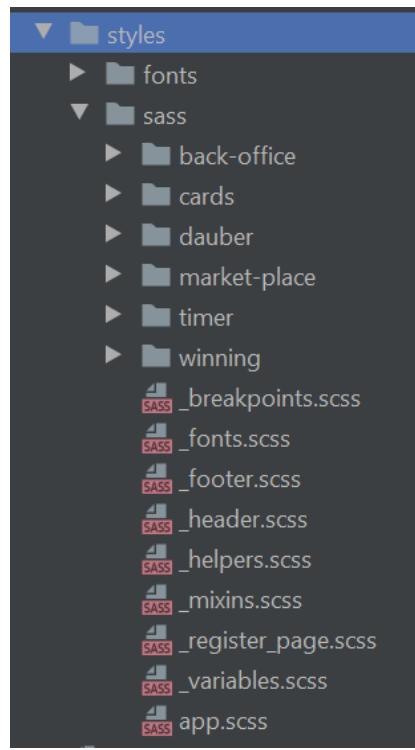
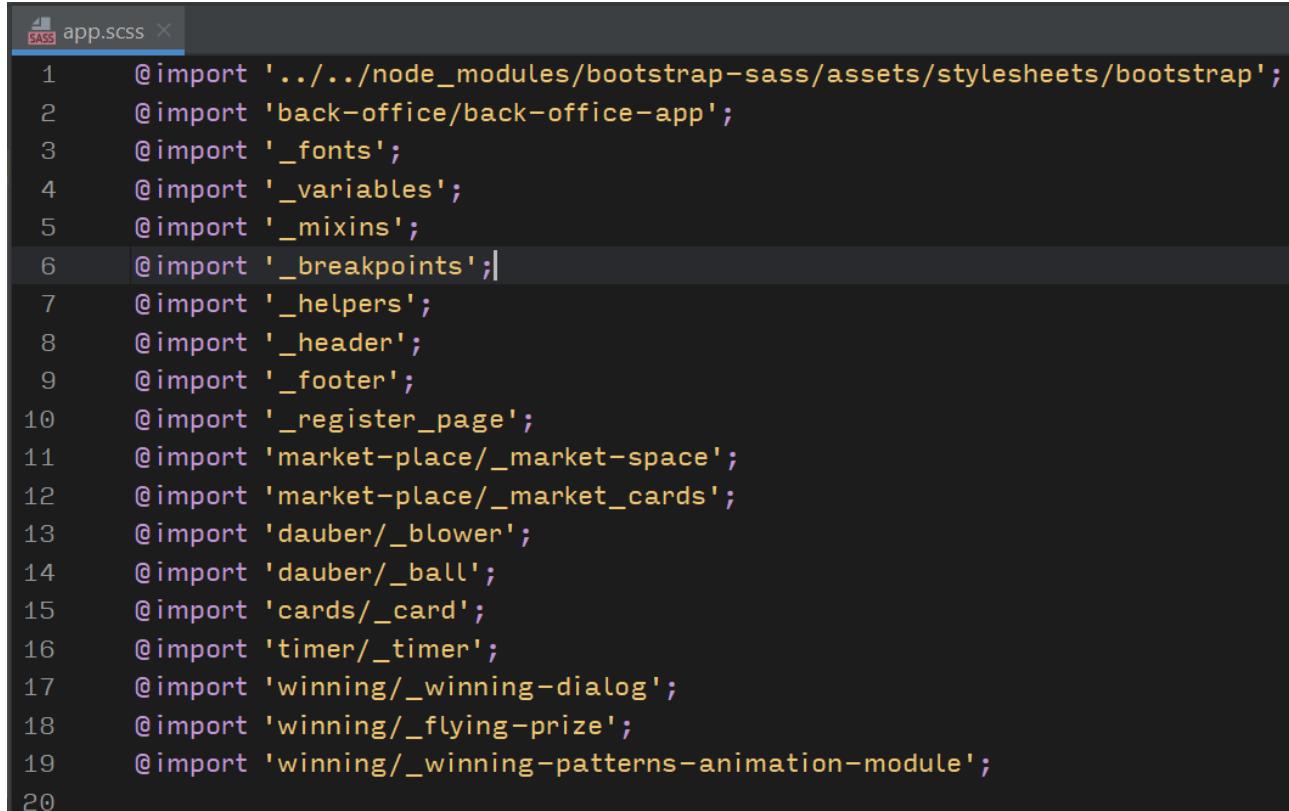


Figure 5: Структура на CSS стиловете на приложението

- d) Стиловете са разпределени по папки, в общия случай, отговарящи за съответен елемент от потребителския интерфейс на приложението. Входна точка е файлът [app.scss](#), в който с помощта на SASS се импортират всички останали файлове със стилове.



```
SASS app.scss ×
1  @import '../../node_modules/bootstrap-sass/assets/stylesheets/bootstrap';
2  @import 'back-office/back-office-app';
3  @import '_fonts';
4  @import '_variables';
5  @import '_mixins';
6  @import '_breakpoints';
7  @import '_helpers';
8  @import '_header';
9  @import '_footer';
10 @import '_register_page';
11 @import 'market-place/_market-space';
12 @import 'market-place/_market_cards';
13 @import 'dauber/_blower';
14 @import 'dauber/_ball';
15 @import 'cards/_card';
16 @import 'timer/_timer';
17 @import 'winning/_winning-dialog';
18 @import 'winning/_flying-prize';
19 @import 'winning/_winning-patterns-animation-module';
20
```

Figure 6: Импортиране на SASS файлове

- e) Този начин на структуриране на стиловете в едно приложение предоставя по-лесна техническа поддръжка във времето и по-бързо ориентиране в проекта на нови разработчици.

3. Tests – функционалността на самата игра и бек-офис приложението е валидирана с юнит тестове. Те са разположени в две директории:

- a) [front-end-tests](#) – разпределението им, както при стиловете, е в общия случай по директории, в които се намират тестовете за компонент или група взаимосвързани (или използвщи функционалността си) компоненти

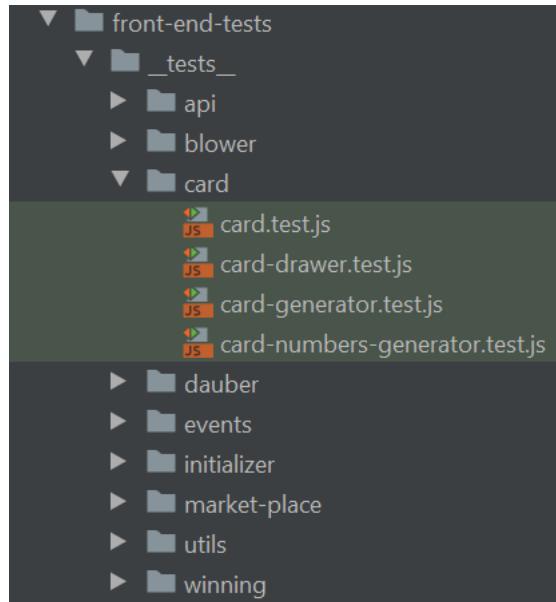


Figure 7: Директорийна структура на фронт-енд тестове

- b) [back-office-tests](#) – структурата на бек-офис тестовете се различава от тези на самата игра, тук е заложен принцип, според който тестваме отделни, модулирани компоненти, с които са изградени страниците и контролите в бек-офис приложението

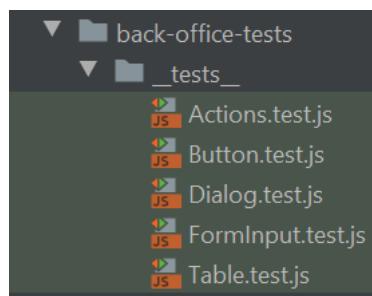


Figure 8: Директорийна структура на бек-офис тестове

E. Графично представяне на структурата на приложението

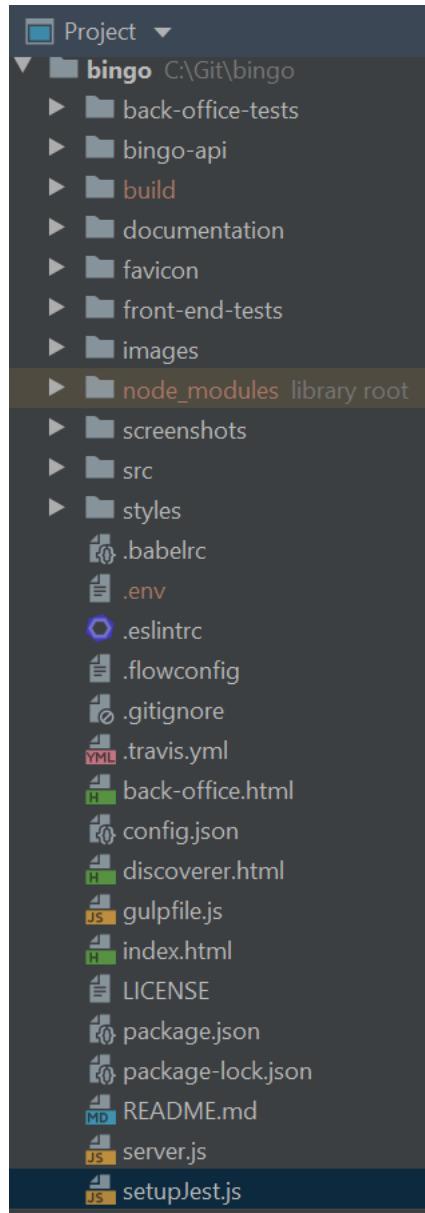


Figure 9: Обща структура на проекта

1. Изводи и бележки достигнати по време на разработката:

- **Преизползваен код** – добра практика е когато е възможно всеки код, който се повтаря, да се модулизира и да бъде преизползван, което води до по-лесно и бързо отстраняване на появили се дефекти. Често срещана практика в по-мащабни приложения е „извеждането“ на общо използвана функционалност в т.нр. *utils*. В повечето случаи това е файл (клас) или директория с такива файлове в проекта, чиято

функционалност е лесно достъпна и се преизползва във всички части на приложението.

- **Компонентно стилизиране** – разпределението на стилове по компоненти дава структурираност и компактност, които водят до по-бързо и лесно добавяне и редактиране, както и до по-ефективно въвеждане на нови хора в проекта и запознаването им (“effective onboarding”).
- **Разделение и самостоятелност на съставните части на приложението** – създаване на отделните компоненти така, че всеки да може да бъде заместван с друг и това заместване да не повлиява на работата на останалите части. Например RESTfull API, което да може да се използва от *всяко фронт-енд приложение*, чрез предоставяне на необходими пътища (routes) за изпълнение на различни функционалности, или пък потребителски интерфейс (user interface), което да може лесно да се интегрира и работи с различни програмни интерфейси (application programming interface – API) без това да влияе на неговата работа. Тук може да се направи препратка към т. нар. *microservices* – подход, при който всеки модул изграждащ дадено приложение се реализира като отделна, самостоятелна *микроуслуга*, която в идеалния случай може да се „извади“ от едно приложение и да се включи в друго, без да има нужда от каквато и да било модификация на самата услуга.
- **Валидиране с юнит тестове** – най-бързата и евтина валидация на функционалността на едно приложение се реализира чрез използването на т. нар. *юнит тестове*. Основната идея зад използването им е, че те дават бърза обратна връзка на разработчика по време на самата разработка на дадена функционалност, което води до производството на по-качествен софтуер, тъй като дефектите се отстраняват още на фаза разработка. Това дава по-голяма сигурност на разработчиците и по-ширака сцена за работа на хората от бизнес отделите. Освен това, аргументацията на разработчиците когато се дискутира дадена промяна или въвеждане на нова функционалност, е с по-голяма тежест.
- **Използване на система за контрол на версии и инструменти (build tools)** за автоматично построяване и изпълнение на всеки от съставните компоненти на приложението – тези неща са практически задължителни в съвременна среда при разработка на

софтуер и ползите, които носят са безбройни. Може би най-важните, които да споменем са пестенето на време и съответно по-бърз процес на разработка.

- **Използване на библиотеки за създаване на потребителски интерфейси** – при построяването на т.нар. *бек-офис (back-office)* приложения (някои ги наричат още *админ панели*) може да се окаже доста полезно да се използва някоя от известните и добре доказани библиотеки за построяване на потребителски интерфейси, каквато е [ReactJS](#) в нашия случай. Тъй като в този тип приложения се използват много UI компоненти, които са или вече заложени в такава библиотека, или лесно и бързо могат да бъдат реализирани. Освен това поради известността и широката публика, на която се радват, разработчиците могат да разчитат, на това че функционалностите, които тези библиотеки предоставят са добре известни. Както и че добрите практики и принципи при разработката на софтуер са заложени в тях. Като допълнителна полза за разработчиците може да се спомене възможността за преизползване на даден такъв компонент в различни приложения, например текстово поле с възможност за валидация на различни входни данни – телефонен номер, имайл адрес и т.н.
- **Отзовчив дизайн и media queries** – при използване на медийни заявки за постигане на отзивчивост в дизайна на приложението е добра практика основните точки (breakpoints) да се изведат на едно място ([_breakpoints.scss](#)) във вид на [SASS миксини](#). Това позволява лесно добавяне на нови и редактирането на съществуващи, тъй като са на едно място. Като същевременно си гарантираме преизползваемост.

VI. Технически характеристики

A. **Обем на приложението** – за определяне на обема на приложението - брой файлове и брой на редове код, е използван инструментът [CLOC \(count lines of code\)](#):

1. JavaScript код – основната игра + админ панел:

a) **Брой файлове: 44**

b) **Брой редове: 2834**

```
c:\Git\bingo (master -> origin) (bingo@1.0.1)
λ cloc src
 44 text files.
 44 unique files.
 14 files ignored.

github.com/AlDanial/cloc v 1.85  T=0.20 s (218.0 files/s, 17114.3 lines/s)
-----
Language           files      blank    comment      code
-----
JavaScript          44        470       150      2834
-----
SUM:                 44        470       150      2834
-----
```

Figure 10: JavaScript код – front end – брой файлове и редове код

2. Юнит тестове – игра:

- a) **Брой файлове: 22**
- b) **Брой редове: 1025**

```
C:\Git\bingo (master -> origin) (bingo@1.0.1)
λ cloc front-end-tests\
  22 text files.
  22 unique files.
  11 files ignored.

github.com/AlDanial/cloc v 1.85  T=0.11 s (199.5 files/s, 11317.9 lines/s)
-----
Language           files      blank     comment      code
-----
JavaScript        22         193       30          1025
-----
SUM:              22         193       30          1025
-----
```

Figure 11: Юнит тестове – front end – брой файлове и редове код

3. Юнит тестове – админ панел:

- a) **Брой файлове: 5**
- b) **Брой редове: 174**

```
C:\Git\bingo (master -> origin) (bingo@1.0.1)
λ cloc back-office-tests\
  5 text files.
  5 unique files.
  2 files ignored.

github.com/AlDanial/cloc v 1.85  T=0.05 s (93.0 files/s, 4149.0 lines/s)
-----
Language           files      blank     comment      code
-----
JavaScript        5          44        5          174
-----
SUM:              5          44        5          174
-----
```

Figure 12: Unit tests – админ панел – брой файлове и редове код

4. Бинго API:

- a) **Брой файлове: 9**
- b) **Брой редове: 396**

```
C:\Git\bingo (master -> origin) (bingo@1.0.1)
λ cloc bingo-api\
  9 text files.
  9 unique files.
  5 files ignored.

github.com/AlDanial/cloc v 1.85  T=0.06 s (154.0 files/s, 7903.4 lines/s)
-----
Language           files      blank    comment      code
-----
JavaScript          9         58        8       396
-----
SUM:               9         58        8       396
-----
```

Figure 13: JavaScript код – server side – брой файлове и редове код

5. Стилове (SASS):

- a) **Брой файлове: 27**
- b) **Брой редове: 989**

```
C:\Git\bingo (master -> origin) (bingo@1.0.1)
λ cloc styles\
  27 text files.
  27 unique files.
  10 files ignored.

github.com/AlDanial/cloc v 1.85  T=0.19 s (139.6 files/s, 5890.6 lines/s)
-----
Language           files      blank    comment      code
-----
Sass              27        140       10      989
-----
SUM:              27        140       10      989
-----
```

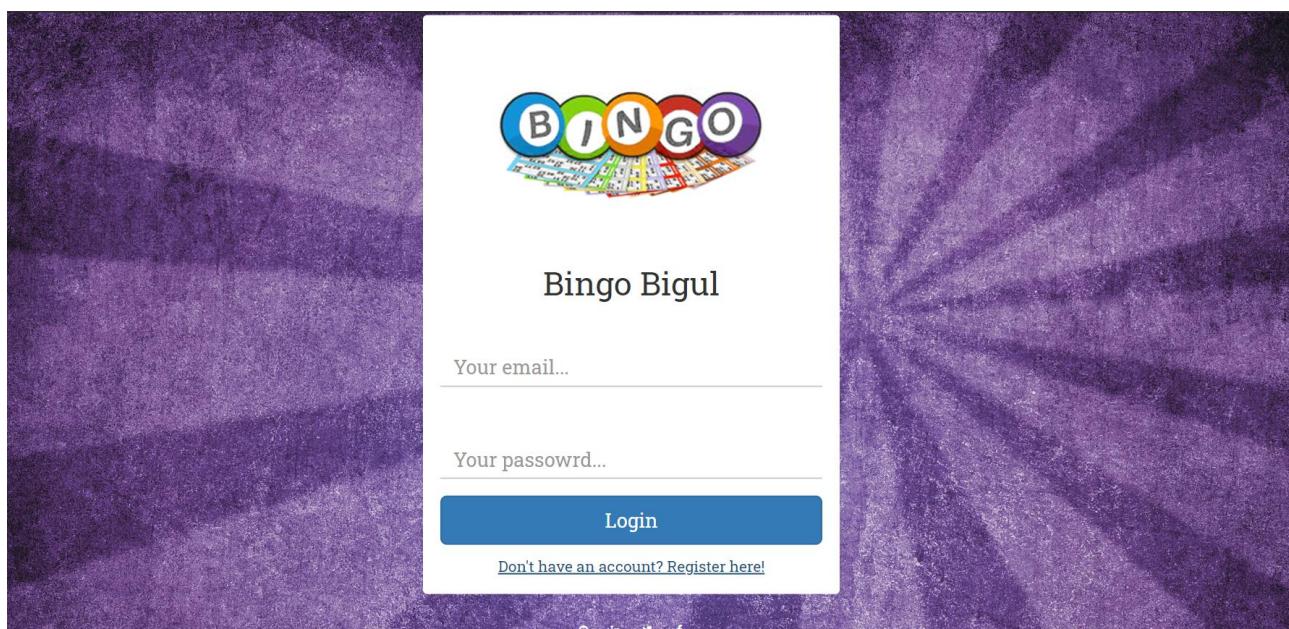
Figure 14: SASS код – брой файлове и редове код

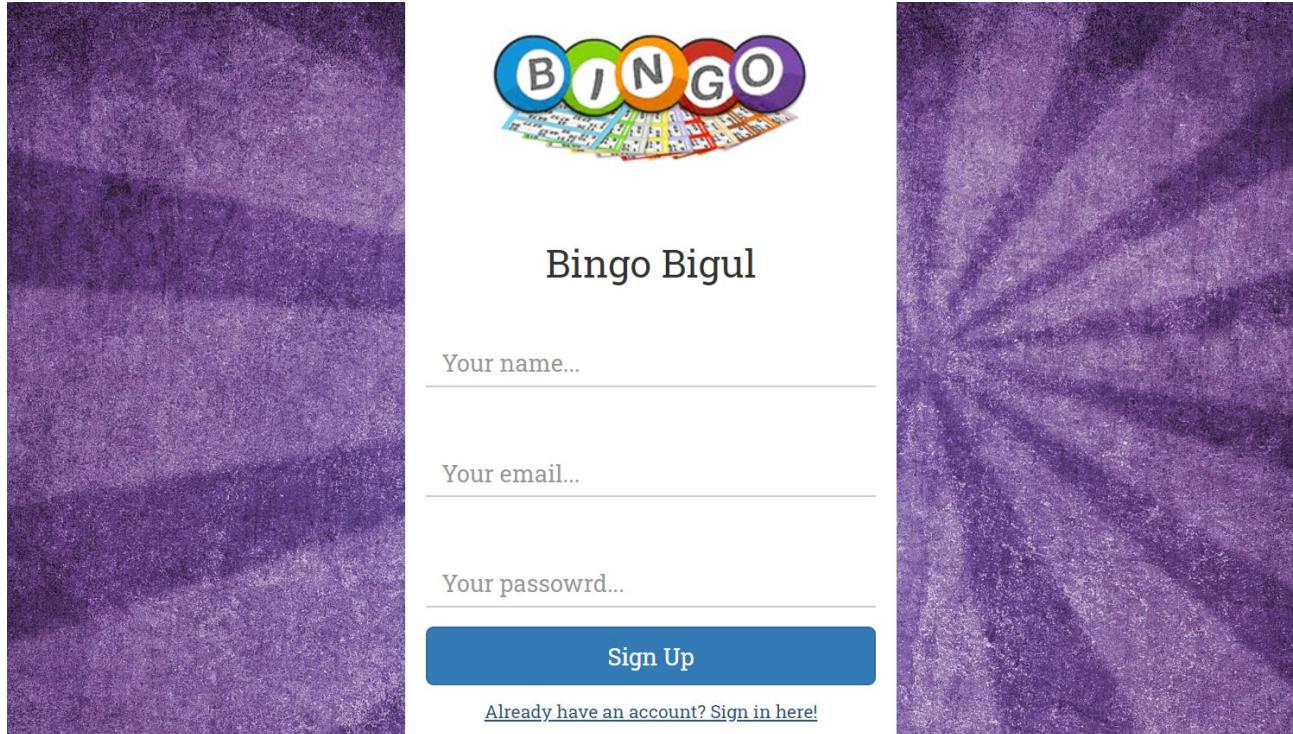
B. *Responsiveness – “отзивчив” дизайн*

1. Реализация – в стилизирането на приложението е заложена идеята за отзивчив дизайн, което означава, че при промяна на размера на устройството, на което се гледа приложението, се променят размера и позицията на елементите. Реализацията на този подход е изпълнена чрез използването на т. нар. [*media queries*](#). Това е способ при писането на CSS стилове, чрез който за определени размери на экрана се задават определени CSS свойства, които „нагаждат“ потребителските интерфейси според наличното пространство. По този начин се постига известна степен на универсалност, тъй като едно и също приложение изглежда добре и е ползваемо на различни по размер устройства. Най-често тези устройства се делят на десктоп компютри, таблети и мобилни телефони, като за всяко от тях е известна резолюцията, на която приложенията се виждат най-добре.

2. Примери:

a) Екрани за вход и регистрация – десктоп





b) Екран за вход и регистрация – мобилен телефон



Bingo Bigul

Bingo Bigul

Your email...

Your name...

Your password...

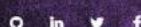
Your email...

Login

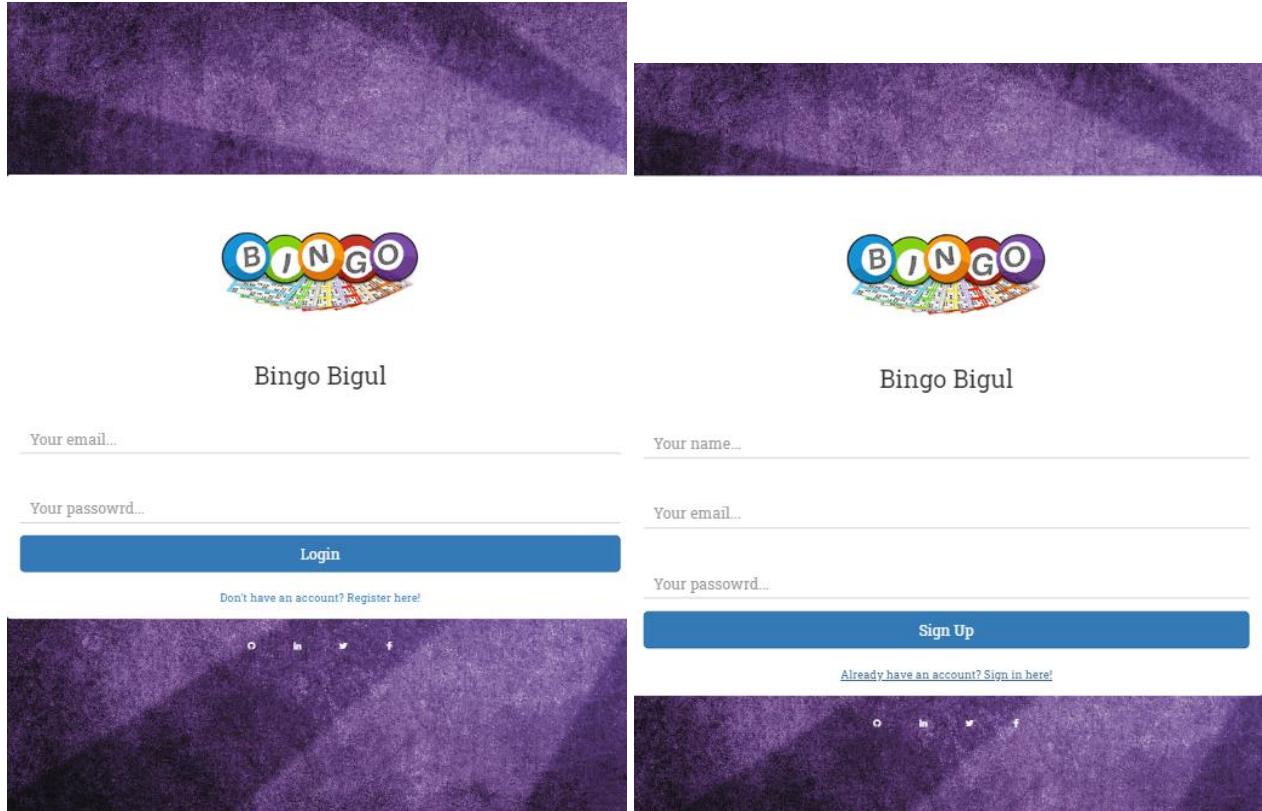
Sign Up

[Don't have an account? Register here!](#)

[Already have an account? Sign in here!](#)



c) Екрани за вход и регистрация - таблет



d) Лоби екран – десктоп

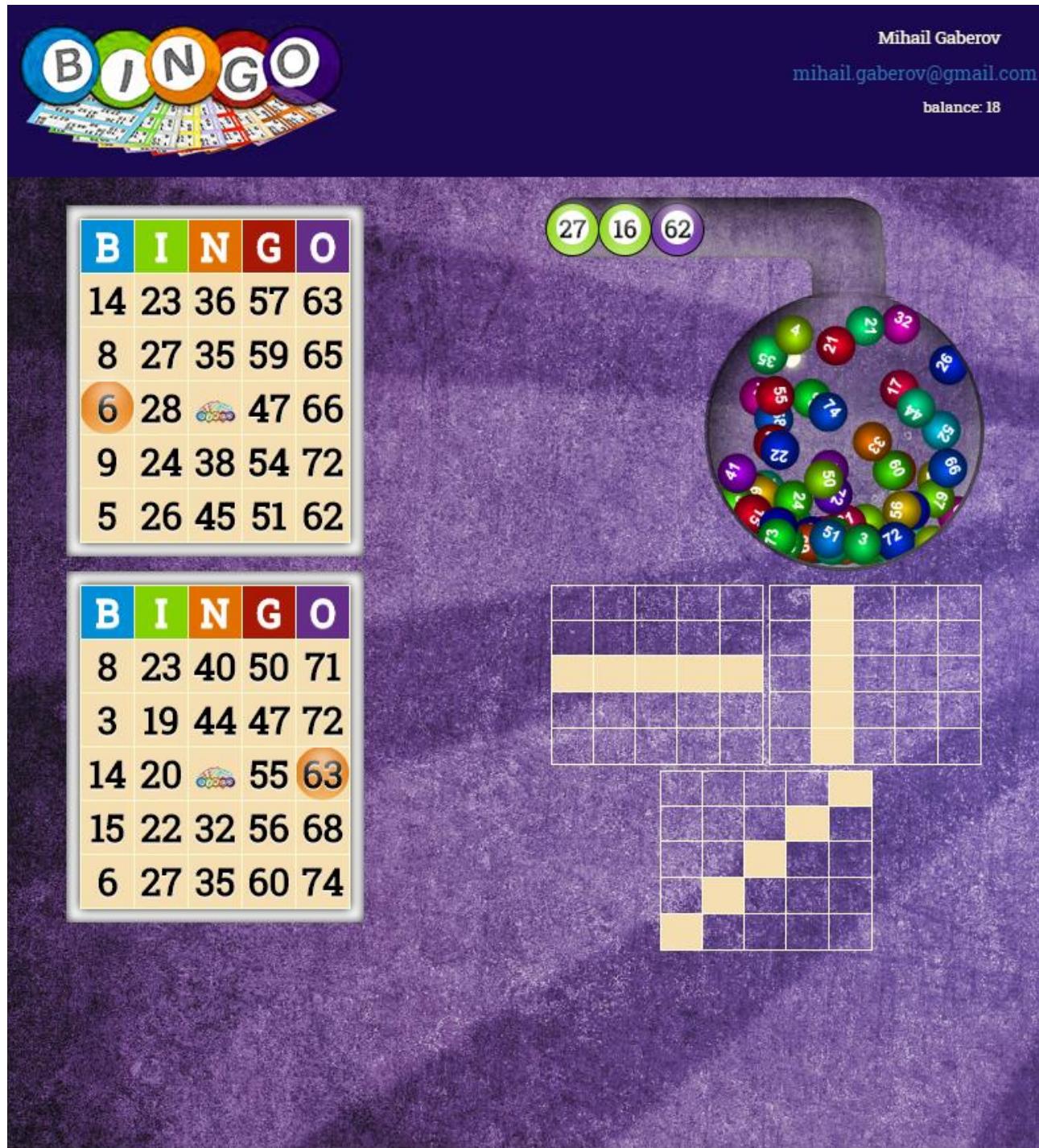
e) Лоби еcran – мобилен телефон



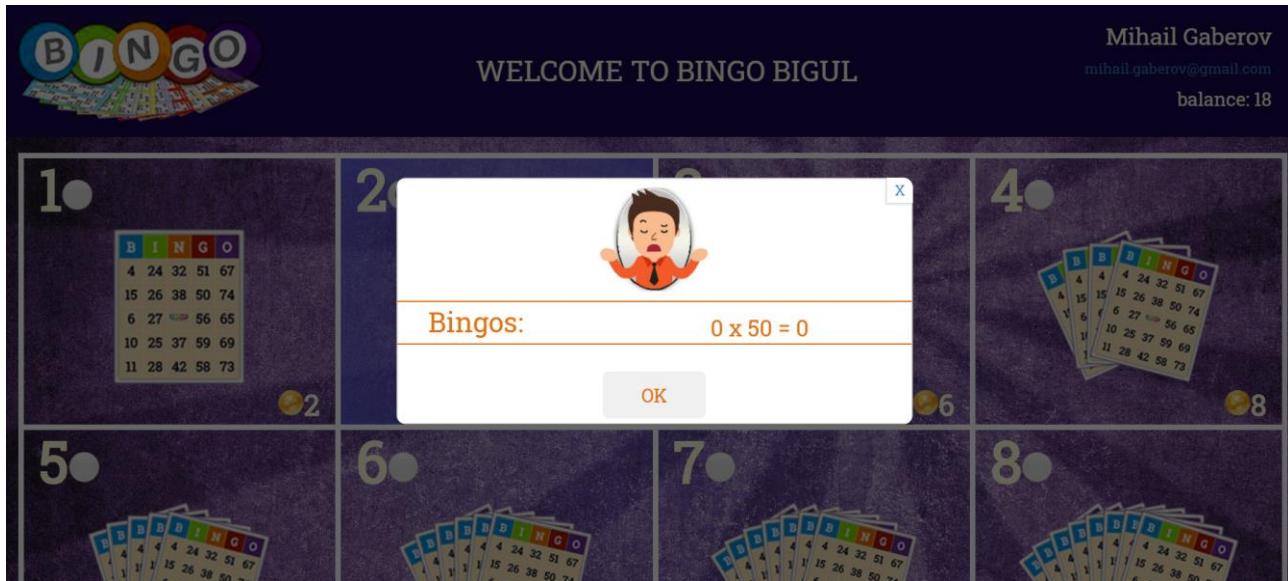
f) Лоби экран – таблет



g) Игрови екран – таблет



h) Край на игра - загуба



i) Бек-офис панел - десктоп

The screenshot shows the 'Bingo Bigul Back Office' application. The top navigation bar includes the logo and the text 'Bingo Bigul Back Office' on the left, and 'Logout' on the right. Below the navigation bar, there's a search bar with the placeholder 'Search 2 records...'. A button labeled '+ add' is located on the left. The main area displays a table with the following data:

Email	Name	Balance	Wins	Actions
mihail.gaberov@gmail.com	Mihail Gaberov	26	0	
mihail.gaberov@zoho.com	Mihail Gaberov	50	0	

j) Бек-офис панел – таблет

The screenshot shows a mobile application interface for the Bingo Bigul Back Office. At the top, there is a purple header bar with the logo 'BINGO' and the text 'Bingo Bigul Back Office'. On the right side of the header is a 'Logout' button. Below the header, there is a purple button labeled '+ add'. To the right of the button is a search bar with the placeholder text 'Search 2 records...'. The main content area is a table with the following columns: Email, Name, Balance, Wins, and Actions. There are two rows of data:

Email	Name	Balance	Wins	Actions
mihail.gaberov@gmail.com	Mihail Gaberov	26	0	
mihail.gaberov@zoho.com	Mihail Gaberov	50	0	

C. Конфигурируеми настройки - както беше споменато по-рано, всички предварителни конфигурации на играта се намират във файла config.json. Те са както следва:

```
"id": "1" // служебен идентификационен номер на играта
"name": "American Bingo" // име на играта
"numbers": [
  1, 2, 3, 4, 5,
  6, 7, 8, 9, 10,
  11, 12, 13, 14, 15,
  16, 17, 18, 19, 20,
  21, 22, 23, 24, 25,
  26, 27, 28, 29, 30,
  31, 32, 33, 34, 35,
  36, 37, 38, 39, 40,
  41, 42, 43, 44, 45,
  46, 47, 48, 49, 50,
  51, 52, 53, 54, 55,
  56, 57, 58, 59, 60,
  61, 62, 63, 64, 65,
  66, 67, 68, 69, 70,
  71, 72, 73, 74, 75
] // числата, които могат да се паднат в една Бинго карта
```

```

"skin": {
    "name": "original" // име на темата за стиловете
},
"appTitle": "Welcome to Bingo Bigul" // мета заглавие на играта
"turnsCount": 23 // брой ходове след които играта свършва
"freeSpotImgPath": "<img src='../../images/small_logo_30x30.png' />" // път до картина, която се появява в средата на бинго картите
"drawIntervalSeconds": 6 // времеви интервал в секунди, през който се теглят новите топки
"beforeStartGameSeconds": 3 // брой секунди, които се отброяват преди старта на нова игра
"marketCards": true // флаг, чрез който се включва или изключва, т.е. става невидим компонента с картите за закупуване в лоби екрана
"dauber": true // флаг, чрез който се включва или изключва, т.е. става невидим компонента за маркиране на числата – „даубер“
"playingCards": true // флаг, чрез който се включва или изключва, т.е. става невидим компонента игрални карти, т.е. тези които се виждат по време на самата игра
"mainGame": true // флаг, чрез който се включва или изключва, т.е. става невидим компонента основна игра
"winningDialog": true // флаг, чрез който се включва или изключва, т.е. става невидим компонента отговарящ за показването на диалоговите прозорци в края на играта
"cardPrice": 2 // цена във виртуална валута за една бинго карта
"winPatternsAnimModule": true // флаг, чрез който се включва или изключва, т.е. става невидим компонента визуализиращ моделите за победа

```

D. Изводи за техническата стойност на приложението:

1. Актуалност на използваните библиотеки – за създаването на приложението са използвани едни от най-популярните и широко разпространени технологии в софтуерната индустрия в наши дни. А именно:

a) **JavaScript** – Програмен език, създаден през 90-те години. В началото на съществуването си се използва главно за валидации на форми и реализирането на кратки анимации, което се счита за доста модерно за времето си. В последствие, след 2000-та година бележи бум в развитието си. На пазара се появяват доста JavaScript базирани библиотеки, като пример може да се спомене една от най-популярните и до днес – [jQuery](#). През 2008г е създаден [Node.js](#) – runtime среда базирана на [Chrome's V8 JavaScript Engine](#), с което популярността и използваемостта на JavaScript се увеличава многократно. На пазара се появяват плетора от библиотеки и инструменти, които се възползват от способността на JavaScript да се изпълнява както от клиентска, така и от сървърна страна. Някои от големите компании в софтуерната индустрия, в лицето на Фейсбук и

Гугъл предприемат стъпки към създаване ([ReactJS](#)) или продължаване на разработката ([AngularJS](#)) на свои такива библиотеки. Една от тези библиотеки за построяване на потребителски интерфейси е използвана в текущата дипломна работа.

- b) **HTML5** – може би най-известното дете в квартала на уеб програмирането, HTML5 е петата версия и текуща версия на HTML (HyperText Markup Language). Маркъп езикът, който днес широко се използва в уеб пространството за дефиниране на свойствата и поведението на уеб съдържание. С други думи за създаване структурата (DOM) и изгледа (с помощта на CSS) на съдържанието. В днешно време това включва не само известните на всички уеб страници, но и приложения (в това число мобилни), в които потребителските интерфейси се създават с помоща на [HTML тагове и атрибути](#) – градивните елементи на езика. Петата версия на езика освен допълнения към маркъпа във вид на нови тагове и атрибути, донесе със себе си няколко програмни интерфейса (APIs), които са заложени в основите на почти всяко съвременно уеб или мобилно приложение. Такъв интерфейс е например [Web Storage](#), който предоставя на приложенията методи и протоколи за съхраняване на данни на страната на клиента, разбирай в браузър или подобен програмен продукт, който разбира се подчинява и изпълнява HTML5 спецификациите.
- c) **ReactJS** – Популярна библиотека за създаване на потребителски интерфейси, разработвана от Фейсбук. Една от най-силните страни на React е изключително голямата аудитория, на която се радва. Което води до доста голямо количество свободно достъпни материали, които помагат за лесната му интеграция и използване. Също така е важно да се спомене за компонентно ориентираната философия на React – в разработките с него всички съставни части на един потребителски интерфейс се обособяват в отделни, самостоятелни компоненти. Метод, който дава на разработчика модуларност и съответно по-лесен и бърз процес на разработка и отстраняване на дефекти. Като този начин на мислене и работа е заложен от самото начало на разработката и ако екипът го следва добре, много скоро след началото на даден проект започва да се радва на

преизползвани компоненти, което намалява времето за разработка и увеличава продуктивността значително.

d) **SASS** – това е един от най-известните и широко застъпени на пазара [CSS препроцесори](#), други такива са [LESS](#) и [Stylus](#). Същността на тези програмни продукти е, че позволяват генерирането на CSS код, чрез синтаксиса предоставен от препроцесора. Тъй като тези синтаксиси носят на програмистите добре познати артефакти от други програмни езици, като например възможността за използване на променливи, функции, миксини и други, това значително улесняват структурирането и употребата на CSS код. Нещо, което носи доста голяма полза когато говорим за големи по размер проекти, с много и различни екипи, участващи в тях. След появата на CSS процесорите, [SASS](#) бидейки един от първите, набира голяма популярност и според едно бързо, неформално проучване в Гугъл, води класацията за най-използвания CSS препроцесор днес. Което беше предпоставка за включването му в текущата разработка.

2. Нужда от допълнително тестване – в бъдещи версии на приложението се планира имплементацията на допълнителни и различни по вид тестове, като средство за по-доброто обезпечаване работата на самата игра и бек-офис приложението:

- a) По-голямо покритие (test coverage) с **unit** тестове
- b) Въвеждане на интеграционни тестове (**integration** tests), които служат за проверка на взаимовръзките между отделните градивни части на приложението (инфраструктура, връзка между различни модули, като например базата данни и бек-енд услугите)
- c) Въвеждане на **end to end** тестове - с помощта на библиотеки като [Cypress.io](#) е възможно да се създадат функционални тестове, с които се тестват функционалностите на приложението, така както крайният потребител ще ги използва. Като пример може да се даде тест на форма за вход в приложението, където са възможни няколко сценария – успешен вход, грешно въведено име или парола и т.н. С инструменти като този тези проверки могат да се

автоматизират и да се използват при всяка промяна на текуща функционалност или при добавяне на нова.

3. Бързодействие:

- a) **Моментно състояние** – от гледна точка на бързодействие моментното състояние на приложението (самата игра плюс бек-офис панела) е напълно задоволително. Това означава, че играчът може свободно да използва личния си компютър или мобилно устройство (таблет, смартфон) за провеждането на една игра, без да усети каквите и да било софтуерни забавления при изпълнението ѝ. Същото важи и за роля администратор, които ще използват админ панела за администриране на регистрираните играчи.
- b) **Места за подобреие** – може би най-слабото място на едно приложение, когато говорим за бързодействие, са анимациите, които се изпълняват в него. В текущата работа за реализирането на анимации, освен чист JavaScript, е използвана библиотеката [PaperJS](#). Постигнатите резултати при анимирането на движението на топките в сферата, както и самото им напускане когато биват изтеглени, са относително приемливи, но също така ни подтикват към обмисляне и планиране на оптимизации в тази насока. Особено когато говорим за мобилни устройства с по-немощни процесори и памет.

VII. Взаимовръзки на отделните подсистеми

A. Клиентска част

1. Структура и основни модули

a) структура

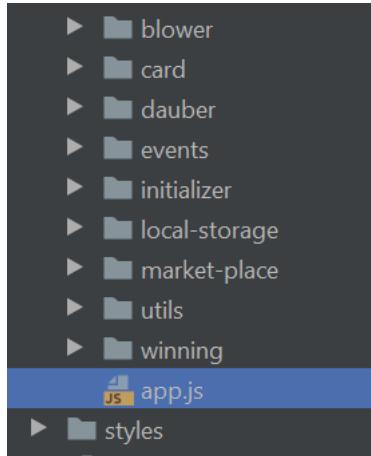


Figure 15: Структура на клиентската част

b) основни модули:

- (1) **App** – стартова точка на играта.
- (2) **Initializer** – предоставя статични методи за добавяне и настройване на основните игрови елементи.
- (3) **Blower** – съдържа класовете, отговарящи за компонентите Ball и Blower. Като тук класът за елемента „топка“ отговаря за тези, които се движат в самата сфера (blower).
- (4) **Card** – съдържа функционалността свързана със създаването и управлението на бинго карти със случайно генеририани числа, необходими за изпълнението на нова игра.
- (5) **Dauber** – съдържа класовете, отговарящи за компоненти Dauber и Ball. Като тук класът за елемента „топка“ отговаря за тези топки, които биват изтеглени и с кратка анимация се подреждат над сферичния компонент.
- (6) **Events** – съдържа имплементация на PubSub модела, както и дефиниция на необходимите игрови събития, като например „изтеглена е нова топка“ или „старт на играта“.

(7) ***LocalStorageService*** – съдържа статични методи за работа с *local storage* – добавяне на данни, проверка на вече записани такива, изтриване при изход от играта.

(8) ***MarketPlace*** – съдържа функционалността за управление на „пазара за бинго карти“, т.е. предоставя методи за закупуване на нови карти, определяне на цената за една карта и проверка на броя избрани за покупка карти.

(9) ***utils*** – съдържа файлове с обща функционалност, която се използва на различни места в приложението. Такива са модулите отговарящи за генериране на случаини числа, анимации, таймери и т.н.

(10) ***winning*** – съдържа файлове с функционалност, отговаряща за действия настъпващи при край на игра – визуализиране на победни диалози, показващи на игрacha спечелените награди, „летяща“ награда – спечелената сума се мести в горния десен ъгъл, където се намира баланса на игрacha и се натрупва към нещо с т. нар. „бангъп“ ефект. Тук се намира и модула, показващ анимирани победните шаблони – вертикална линия, хоризонтална линия и двата диагонала.

2. Връзки с другите подсистеми – всички връзки на фронт-енд частта с останалите части на приложението се извършват чрез няколко основни модула, намиращи се в директорията `src/API`:

- a) **api-consts** – съдържа дефиниция на константен обект с всички пътища (routes), които се използват за интеракция с бек-енд услугите.
- b) **api-controller** – съдържа методи за изпълнения на основни действия в приложението, като например вход на играч или администратор, обновяване на игровия баланс и т.н. Използва функционалност предоставена от други модули за тази цел (например db-service).
- c) **db-service** – съдържа методи за работа с базата данни чрез предоставените от бек-енд частта услуги – четене, запис и изтриване на данни за играчите и администраторите. Действия като нова регистрация, вход и аутентикация се извършват през този модул.

B. *Back-office панел*

1. Структура и основни модули

- a) структура

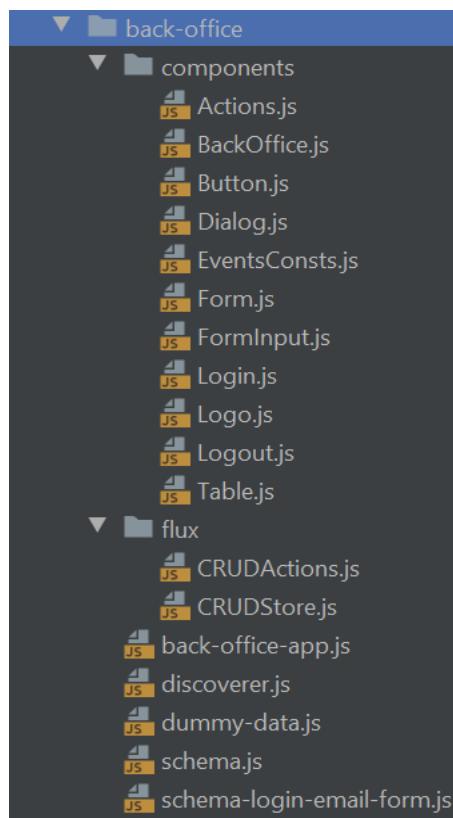
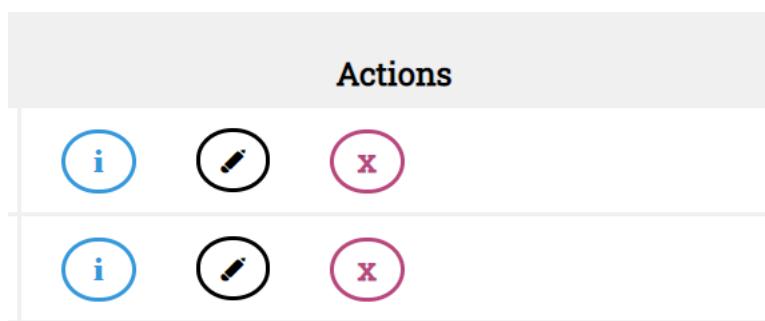


Figure 16: Структура на бек-офис панела

b) основни модули – функционалността, изграждаща бекофис приложението е разположена в директория `scr/back-office`, като в поддиректория `components` се намират основните градивни компоненти на потребителския интерфейс (бутони, диалози и т.н.). Тъй като за създаване на приложението е използван ReactJS с [Flux](#) от Фейсбук, в папка `flux` се намират файловете отговарящи за управление на състоянието на приложението (application state).

Гореспоменатите компоненти са както следва:

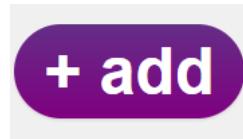
(1) **Actions** – предоставя контроли за преглеждане на информацията за даден регистриран потребител, за нейното редактиране или за напълното му изтриване.



(2) **BackOffice** – основен компонент за работата на бекофис панела, представлява контролите, които администраторите виждат след успешен вход в админ панела. Използва останалите модули (бутони, диалози) за изграждане на потребителския интерфейс, необходим на администраторите за успешно изпълняване на действия като добавяне на нов играч, изтриване на съществуващ или редактиране на информация за такъв.

Actions					
Email	Name	Balance	Wins		
mihail.gaberov@gmail.com	Mihail Gaberov	16	0		
mihail.gaberov@zoho.com	Mihail Gaberov	50	0		

(3) **Button** – компонент бутон, предоставя функционалност за динамично дефиниране на текста, който се показва като етикет на бутона, както и т.нар. *onClick hanlder* – функция, която се изпълнява при натискане на бутона.



(4) **Dialog** – компонент диалогов процесор, използва се при случаите когато се изисква потвърждение или отказ от потребителя, обикновено при извършване на действия като изтряване на съществуващ играч. Позволява динамично задаване на модалността на диалога, т.е. дали да бъде възможно кликането извън него или не.



Are you sure you want to delete
"Mihail Gaberov"?

[Cancel](#) [Delete](#)

(5) **EventConsts** – съдържа константен обект с имената на събитията, които се използват от приложението.

(6) **Form** – компонент Форма, предоставя текстови полета и контроли за попълване и редактиране на данни.

Email:

mihail.gaberov@gmail.com

Name:

Mihail Gaberov

Password:

Balance:

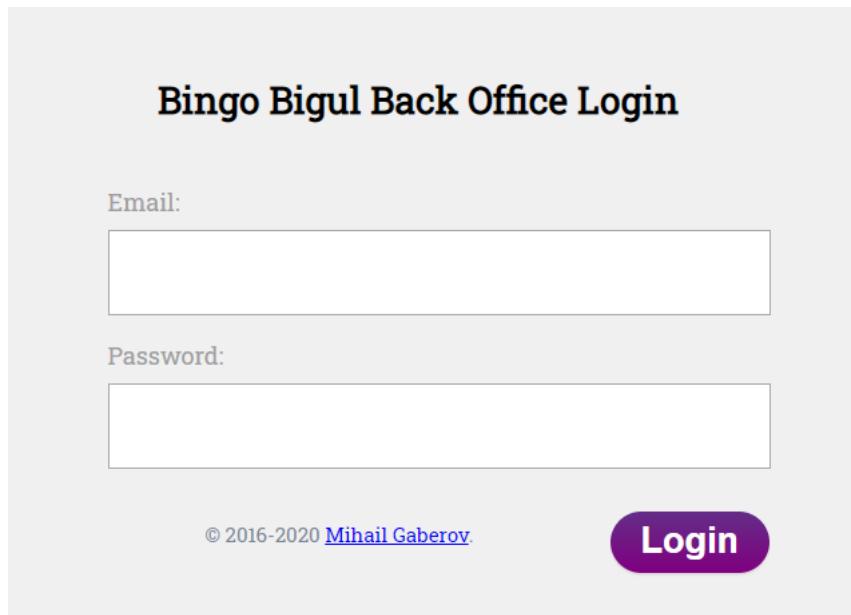
16

(7) **FormInput** – компонент, предоставяещ различни типове текстови полета – такива за въвеждане на имейл адрес, на пароли, на чисто числови или чисто текстови стойности.

Email:

mihail.gaberov@gmail.com

(8) **Login** – компонент, отговарящ за вход екрана към админ панела.



(9) **Logo** – компонент, визуализиращ логото в горния ляв ъгъл на экрана.



(10) **Logout** – компонент, отговарящ за визуализирането на бутона за изход и функционалността обвързана с това – „слушане“ за събития и консумиране на метода за изход от Flux модулите ([CRUDStore](#)).



(11) **Table** – компонент, отговарящ за визуализирането основната таблица в админ панела, включително бутооните за извършване на действия върху данните.

Email	Name	Balance	Wins	Actions		
mihail.gaberov@gmail.com	Mihail Gaberov	16	0			
mihail.gaberov@zoho.com	Mihail Gaberov	50	0			

2. Връзки с другите подсистеми – админ панел приложението осъществява взаимодействията си с дугите части на системата чрез модулите [ApiController](#) и [CRUDStore](#).

C. API

1. Структура и основни модули

a) структура

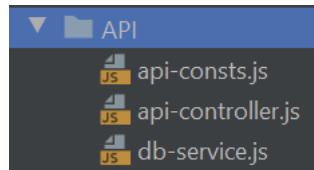


Figure 17: Структура на API модула

- b) основни модули – градивните части на програмния интерфейс осигуряващ връзка между фронт-енд и бек-енд услугите се намират в API директорията, описани в т. А.2.
2. Връзки с другите подсистеми – услугите, които API модулът предоставя сами по себе си играят ролята на връзки, които спомагат за взаимодействието на различните части (front end, back end, database) на приложението.

D. *Discoverer*

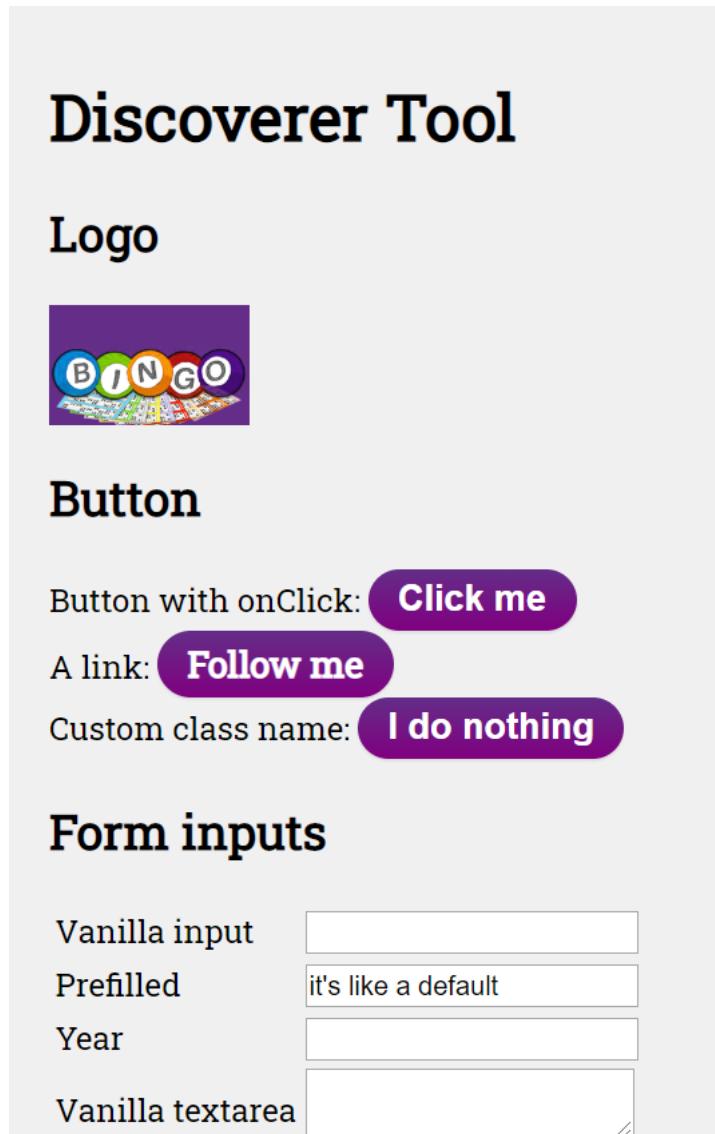


Figure 18: Инструмент Discoverer

1. Описание – т.нар. инструмент „дискавърър“ служи за място където да се добавят ново разработени компоненти, чиято цел е да бъдат преизползвани. Така, в по-голям екип от разработчици, или в проект, в който участват няколко екипа, се осигурява консистентност в работата им и се избягва повтаряемостта. Тоест,

една и съща по вид, работа да бъде извършвана от няколко человека, без те дори да знаят за това. Същото важи и за ситуации където екипи работят по различни проекти, но ползват обща базова функционалност и програмни компоненти. Често срещан в практиката пример е случая когато два различни екипа имат нужда от, да кажем, компонент „диалог“. Всеки от тях прави собствена разработка, която се припокрива като функционалност с тази на другия екип. Точно в такива ситуации инструмент като този (в днешно време в сферата на разработване на потребителски интерфейси набра популярност подобен такъв, който се нарича [storybook](#)) бива доста полезен. Обикновено процедурата при работа с него е следната – когато екип има нужда от нов компонент, първо проверява в *discoverer* дали вече съществува такъв и съответно го преизползва. Ако все още няма създаден такъв компонент, екипът го създава, добавя го в *discoverer* и го документира. Като всяка заинтересована страна трябва да научава за тези промени. Различните компании използват различни комуникационни средства за тази цел – имайл нотификации от самия инструмент, [confluence](#) страници, документацията на самото GitHub репозитории или просто [Slack](#) канал, предназначен само за такъв тип съобщения.

2. Елементи – в проекта са използвани следните елементи:

a) Лого



b) Бутон



c) Текстови полета

Form inputs

Vanilla input	<input type="text"/>
Prefilled	<input type="text" value="it's like a default"/>
Year	<input type="text"/>
Vanilla textarea	<input type="text"/>

d) Форма

Form

Email:	<input type="text" value="mihail.gaberov@gmail.com"/>
Name:	<input type="text" value="Mihail Gaberov"/>
Password:	<input type="password" value="*****"/>
Balance:	<input type="text" value="50"/>
Wins:	<input type="text" value="20"/>

e) Форма само за четене (read only)

Form readonly

Email:
mihail.gaberov@gmail.com

Name:
Mihail Gaberov

Balance:
555

Wins:
1232

f) Бутони за действия (actions)

Actions



g) Диалогов прозорец (dialog) – предлага възможност за промяна на бутоните

Dialog

Out of the box example

Hello, dialog!

[Cancel](#) [OK](#)

No cancel, custom button

Anything goes here, see: [A button](#)

[Whatever](#)

VIII. Реализация

A. Имплементация

1. Общи

а) Класове – приложението разполага със следните класове:

- (1) *ApiController*
- (2) *DbService*
- (3) *Ball*
- (4) *Blower*
- (5) *Card*
- (6) *CardDrawer*
- (7) *CardGenerator*
- (8) *CardNumbersGenerator*
- (9) *Dauber*
- (10) *PubSubService*
- (11) *Initializer*
- (12) *LocalStorageService*
- (13) *MarketCards*
- (14) *Animator*
- (15) *Bangup*
- (16) *Timer*
- (17) *ViewManipulator*
- (18) *Flying Prize*
- (19) *WinPatternsAnimModule*
- (20) *WinningDialog*
- (21) *App*

b) Последователност на работа (Workflow):

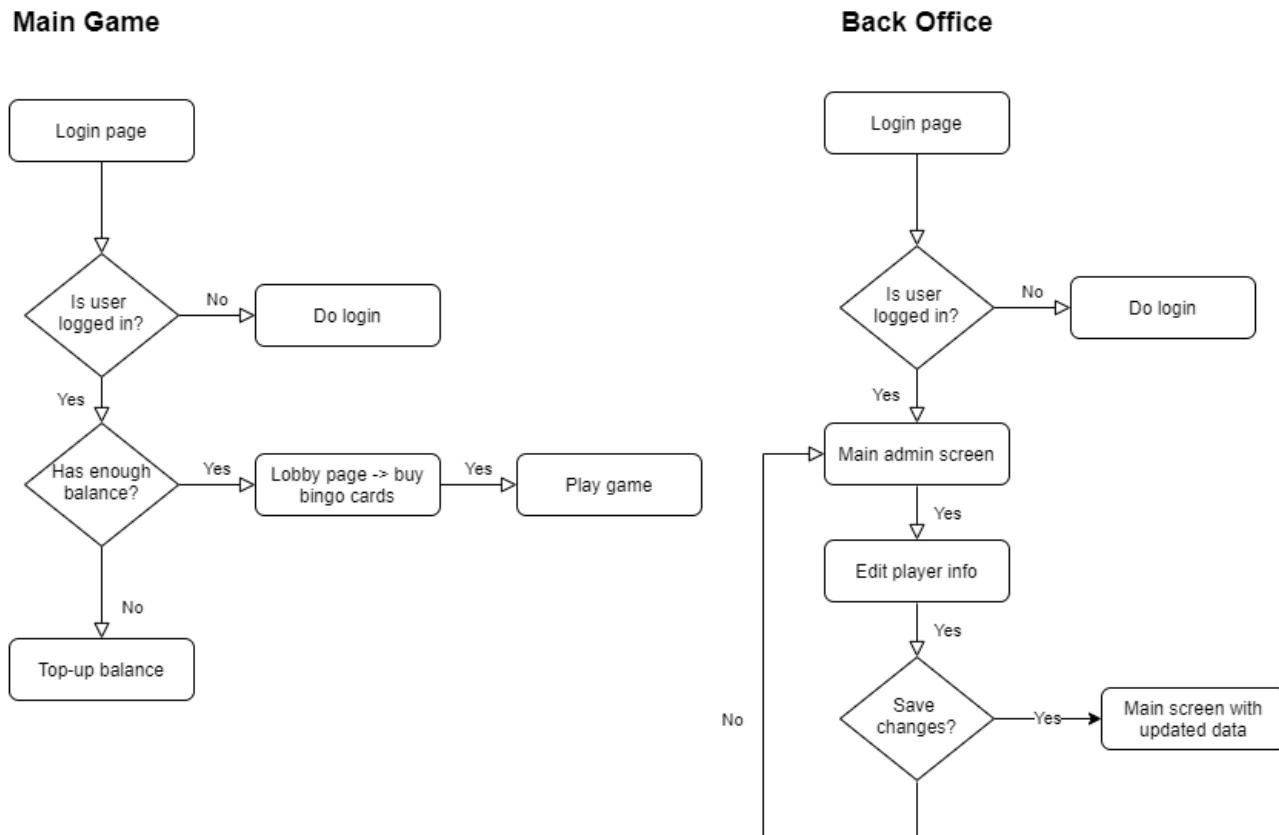


Figure 19: Диаграми, показващи последователността на работа на основната игра и на бек-офис приложението

c) **Utils** – това е общо название за функционалности, които в никаква степен са или биха могли да бъдат общи за проекта и да се преизползват на различни места. Ето какво е определено да се счита за такава функционалност в текущия проект:

- (1) *Animator* – съдържа логика за изпълнение на различни по вид анимации, което се постига чрез използването на различни математически изчисления. Например линейна, квадратична или т.нар. „подскачаща“ (от англ. Bounce) анимации.
- (2) *Bangup* – съдържа функционалност за постигане на т.нар. „бангъп“ ефект – постепено натрупване на сума от една стойност до друга стойност.
- (3) *NumbersGenerator* - съдържа функционалност за генериране на случаини числа в диапазон със зададена минимална и максимална стойност.

- (4) *Timer* - съдържа функционалност за реализиране на таймер за отброяване на зададен брой секунди.
Разполага с възможност за определяне на събитие, което да се стартира след изтичане на зададеното време.
- (5) *Utils* - съдържа методи за преизползване. Например такъв за елиминиране на дублираните стойности в масив или за броене на това колко пъти даден елемент се среща в масив.
- (6) *ViewManipulator* - съдържа функционалност за манипулиране на видими части от приложението, като например показване или скриване на картите за игра, съобщения за грешки, информацията за играта и т.н.
- (7) *WinningPatterns* - съдържа функционалност, която отговаря за проверката дали даден модел за победа е изпълнен – линия по хоризонталата или вертикалата, ляв или десен диагонал или четирите ъгъла.
- d) **Диалози** – в играта са реализирани игрови елементи диалогови прозорци, които се появяват в края на всяка игра, показвайки на играла резултатите от изиграната игра.

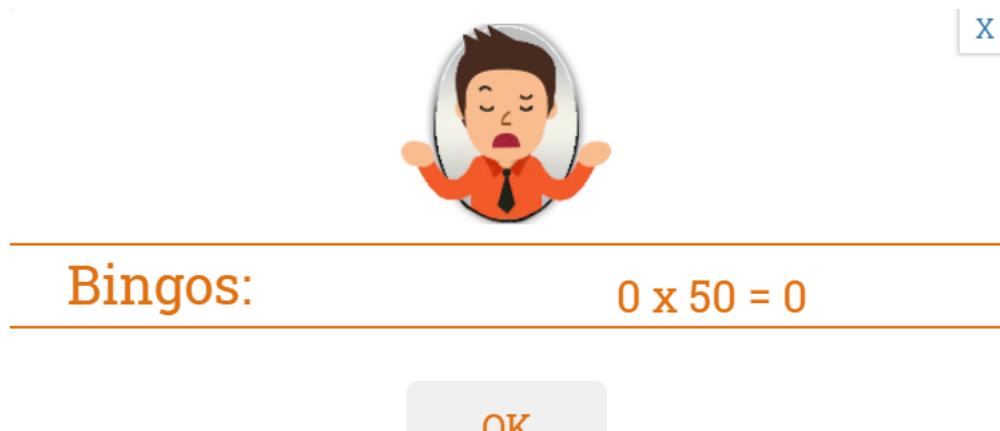


Figure 20: Диалогов прозорец, показващ край на игра без спечелени Бинго карти



Figure 21: Диалогов прозорец, показващ края на игра с три спечелени Бинго карти

e) **Анимации** – целта на производителите на подобен тип игри е да ги правят все по-привлекателни за играчите и един от начините за това е въвеждането на анимации и анимирани елементи тъм където е възможно. В текущата работа анимациите са силно застъпени, като за някои от компонентите е използвана външна, JavaScript базирана, библиотека наречена [PaperJS](#). За други анимации, като например тези в победните диалози е използван само CSS, а за трети, като например излизането и подреждането на изтеглените топки – комбинация между JavaScript и CSS.

f) **База данни** – приложението използва MongoDB – система за обработване на бази данни от документи и колекции от документи. Името на текущата база данни е `bingo-bigul`. Разполага със следните колекции и документи:

(1) **Колекции admins и users**

Collection Name ^	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	
admins	1	266.0 B	266.0 B	2	32.0 KB	
users	2	296.5 B	593.0 B	2	64.0 KB	

(2) **Документи към колекция admins:**

```
_id: ObjectId("5e03162c1b02852d1c649ef4")
name: "Mihail"
email: "mihail.gaberov@gmail.com"
salt: "b243ade4ea2824ec2e65f362a1085b04"
hash: "4ee7998fd740d6f54b241a8ea73ee313ead62c19e8a0b7e138565ff6021757e1bd1e00..."
__v: 0
```

(3) Документи към колекция users

```
_id: ObjectId("5c04d35f8258bd38d0f7c883")
name: "Mihail Gaberov"
email: "mihail.gaberov@gmail.com"
balance: 624
wins: 0
salt: "ff2206a305402742f61ecab5270b07f4"
hash: "f7afffd9d79de98d0807971502f2af69f56debbca8fdb9c18a2e1863af013db91849bbb..."
__v: 0
```

```
_id: ObjectId("5e197ac1f18f6330685ab5f0")
name: "Mihail Gaberov"
email: "mihail.gaberov@zoho.com"
balance: 50
wins: 0
salt: "416e2ed0d9f07ac756b1562cdb839cf8"
hash: "e8aca29fbe7836254e79c33785f6a23ee951fef88a352743800c712998a2c51112d45e..."
__v: 0
```

g) API – услугите, предоставени от бек-енд частта, чрез които играчите и администраторите извършват успешно своите действия:

- **Игра:**

- /setNewBalance – използва се за задаване на нова стойност на баланса на играта
- /profile – използва се за четене на информацията за профила на играта
- /register – използва се за регистрация на нови играчи
- /login – използва се за вход на играта
- /setWins – използва се когато играча спечели нова игра и броя на победите му трябва да се обнови

- **Бек-офис:**

- /getPlayersData – използва се за четене на данните за всички играчи
- /createPlayer – използва се за създаване на нов играч през бек-офис приложението, може да използва за прескачане на нуждата да се направи нова регистрация през самата игра
- /deletePlayer – използва се за изтриване на играч от системата
- /updatePlayerData – използва се за обновяване на данни за играч

- e. /loginAdmin – използва се за вход на потребители с роля администратор
- f. /registerAdmin * - от гледна точка на сигурността, тази опция за момента се включва ръчно от разработчиците, използва се за регистрация на нови администратори на приложението

2. Функционалност на основната игра – функционалността на основната игра може да се раздели условно следните стъпки:

- a) Екран за вход или нова регистрация на играч
- b) Лоби еcran, където играча може да закупи нови карти за игра
- c) Същинска игра, където играча отбелязва изтеглените числа при съвпадение с тези в закупените му карти
- d) Край на игра и обновяване на баланса
- e) Изход от играта

3. Функционалност на приложението за администриране на потребителски данни „бек-офис“:

- a) Екран за вход
- b) Основен еcran с контроли за добавяне, редактиране и триене на данни за играчите
- c) Диалози за добавяне на нови играчи и за редактиране информацията за текущи
- d) Поле за търсене, което филтрира основната таблица с играчите
- e) Изход от приложението

B. Валидация на приложението

1. Основни положения – Валидацията на текущата работа е сравнително опростен процес, в който трябва да бъде проверена правилната работа на основните части на приложението – основна игра и бек-офис приложение. Тази проверка се извършва обикновено от тест екип или специалист по качеството на софтуер (QA). Той трябва да премине през всички очевидни начини за употреба на

приложението – например да провери какво ще се случи при бавна интернет връзка или пълна липса на такава. Възможно ли е играчът да започне нова игра в тези случаи или какъв ще е изходът от вече започната игра.

При разработка на мобилни приложения или такива, предназначени за всякакви устройства – десктоп и мобилни, се извършват редица тестове на устройства с различни по големина екрани и операционни системи (Android и iOS в общия случай). Тъй като в текущото приложение е предвиден „отзовчив“ дизайн, което означава, че е възможно използването му на различни устройства и размери на екрани, тест инженерите трябва да включат тестове на такива устройства в процеса по валидацията.

2. Препоръки

- a) Автоматизация - добра практика при тестването на софтуер е въвеждането на автоматизации. Ако нещо може да се автоматизира, то трябва да се автоматизира. Това спестява огромни усилия на тестовите екипи и води до значително намаляване на времето необходимо за извършване на валидация при пускане на нови версии.
- b) Проверка на основния сценарий (**happy path**) – при всяка валидация на приложение като минимум трябва да се провери основния сценарий на употреба на приложението. В нашия случай това представлява успешна регистрация и логин на играча, и изиграването на една бинго игра. Това трябва да може да се случи само при наличие на достатъчна сума по баланса за закупуване на бинго карти.
- c) Проверка на крайни случаи (**edge cases**) – при всяка валидация на приложение трябва да бъдат покрити крайните случаи при употреба на приложението. Това са проблеми или ситуации, които обикновено не се случват при нормален начин на употреба на приложението. Друго наименование, с което се срещат в софтуерното инженерство е *corner cases*. Те настъпват когато входните данни, от които би имало нужда едно приложение, за да работи нормално, са с екстремни стойности – минимуми или максимуми например. Или такива, които биха изисквали специална обработка. Такива случаи могат да бъдат очаквани или неочеквани. Процесът по планирането и адресирането им може да се окаже доста тежка задача. В нашия случай входните точки за въвеждане

на данни от потребителя към приложението са формите за вход и регистрация и лоби екрана с възможността за закупуване на бинго карти. Следователно там са местата където би било нужно да се направи такава валидация от тест инженерите и от самите разработчици.

C. Ръководство за инсталиране

1. [Node.js](#) – може да се свали и инсталира различните операционни системи (Windows, macOS, Linux) от [тук](#). Предоставя платформа, на която инструментите за построяване на приложението функционират.
2. [NPM modules](#) – това са софтуерните пакети, които се инсталират с помоха на пакетен мениджър наречен *npm* (*Node package manager*). *Инсталирането им се извършва посредством команда `npm install`* (`npm i` е съкратен вариант на същата), която се изпълнява в основната директория на проекта в терминал приложение (*cmd*). Тази команда ще инсталира всички пакети описани в [package.json файла](#):



The screenshot shows a Windows Command Prompt window titled "cmd (Admin)". The command line displays the following text:

```
C:\Users\DeliyuHaiduka
λ cd \git\bingo\

C:\Git\bingo (master -> origin) (bingo@1.0.1)
λ npm install|
```

The window has standard Windows UI elements like minimize, maximize, and close buttons at the top right. At the bottom, there's a taskbar with icons for File Explorer, Task View, and others. The title bar says "cmd.exe".

Figure 22: Команда `npm install`

3. [Gulp.js](#) – инсталирайте глобално инструмента за автоматизиране на билд процеса Gulp със следната команда:

```
npm install gulp -g
```

Бележка: Gulp трябва да бъде инсталиран глобално, което правим като използваме флаг -g, но и локална версия също ще бъде инсталирана, за да се подсигури съвместимостта с версията използвана в проекта.

4. Добавете .env файл в основната директория на проекта (в случай, че все още нямате), който съдържа следните записи:

```
DB_SECRET=<паролата за вашата база данни>
DB_URL=mongodb://127.0.0.1/<името на вашата база данни>
```

5. [MongoDB](#) – свалете и инсталирайте [MongoDB Community Server](#) и се уверете, че можете да използвате [MongoDB shell version v4.0.0](#)

6. Стаптирайте сървъра за базата данни със следната команда:

```
mongod
```

Забележка: пътя по подразбиране, 'dbpath', за MongoDB е '/data/db'. Това означава, че ако инсталирате MongoDB във вашия C:\ диск, когато стартирате командата 'mongod', той ще бъде използван без да е нужно изрично да се подава като параметър. Ако сте инсталирали базата на различен диск, то тогава ще трябва да подадете този параметър изрично, което става по следния начин: '--dbpath=/path/to/your/db'. Ето как би изглеждал реален пример, в който базата е инсталирана в диск D:\|. Повече информация по темата можете да прочетете [тук](#).

След като имате стартирана базата данни е време да стартираме игровия сървър. Това става с изпълнението на следната команда в главната директория на проекта:

```
node server.js
```

7. За да стартирате приложението, използвайте следната команда:

```
gulp
```

8. За да стартирате само играта и юнит тестовете за нея, стартирайте следната команда:

```
gulp fe
```

9. За да стартирате само бек-офис приложението и юнит тестовете за него, стартирайте следната команда:

gulp bo

10. Отворете в браузъра си адрес <http://localhost:8000> и се насладете на играта!

D. Ръководство за потребителя

1. Регистрация – за да има възможност за игра, потребителят трябва да бъде регистриран в системата. Това става чрез коректно попълване на формата за регистрация, която се достъпва от основния екран чрез линка „[Don't have an account? Register here!](#)“.

Самата форма е доста опростена, тъй като изисква от потребителя да попълни само своето име, имейл адрес и парола.



Bingo Bigul

Your name...

Your email...

Your password...

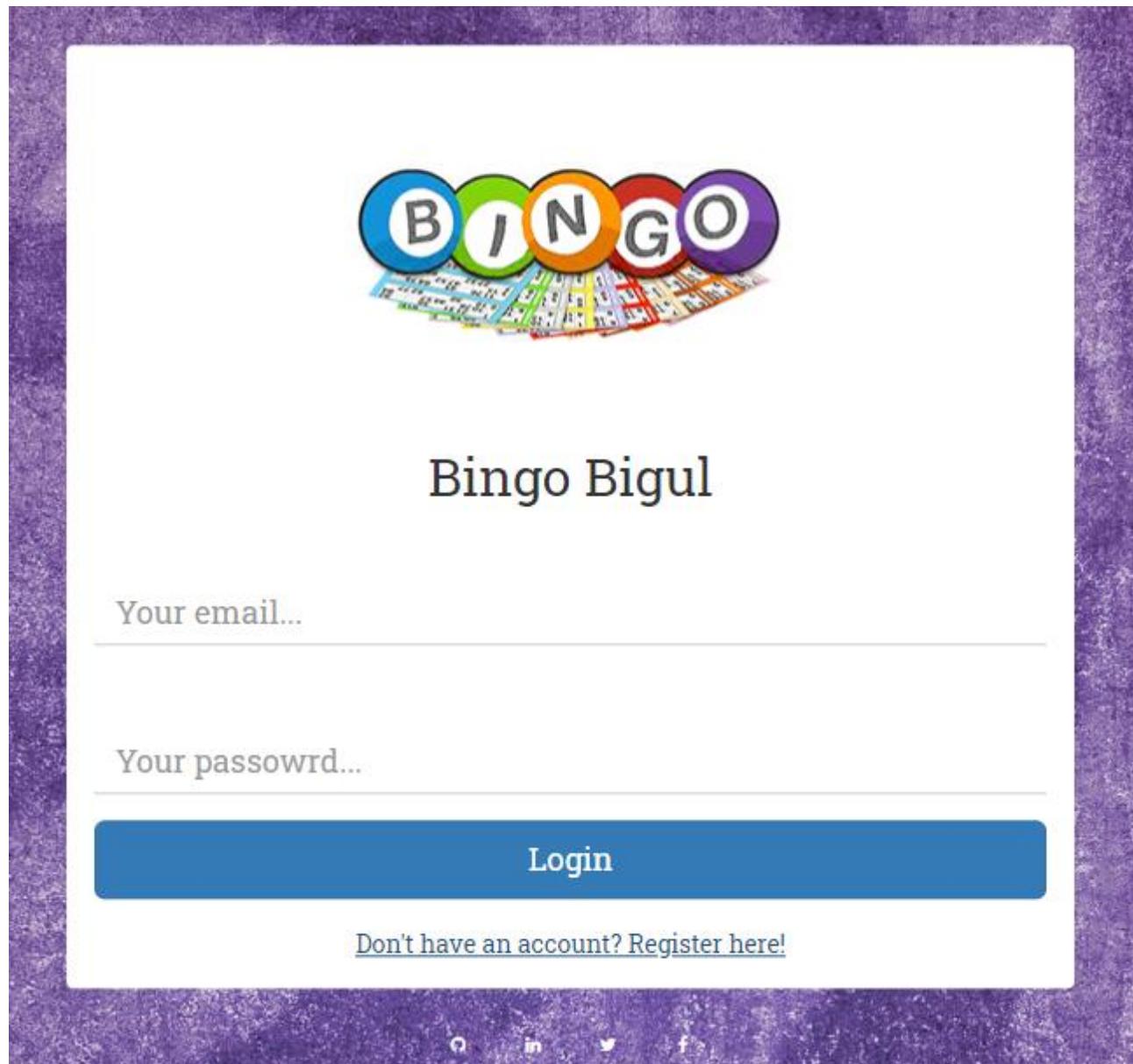
Sign Up

Already have an account? [Sign in here!](#)

Follow us: 

Figure 23: Форма за регистрация

2. Вход (login) – за да има възможност за игра, след успешна регистрация, е необходимо потребителят да се идентифицира в системата, използвайки имейл адрес и парола, въведени при регистрацията.



The image shows a login form for the website Bingo Bigul. At the top center is a logo featuring the word "BINGO" in large, stylized letters, each contained within a colored circle (blue, green, orange, red, purple). Below the logo is the website name "Bingo Bigul". The form itself has two input fields: one for "Your email..." and one for "Your password...". A large blue "Login" button is positioned below these fields. At the bottom of the form, there is a link "Don't have an account? Register here!". Below the form, there are social media sharing icons for Google+, LinkedIn, Twitter, and Facebook.

Your email...

Your password...

Login

[Don't have an account? Register here!](#)

Figure 24: Форма за вход

3. Закупуване на бинго карти – след успешен вход, на потребителя е представен лоби екран, чрез който е възможно закупуването на бинго карти. За целта играчът трябва да разполага с достатъчна сума по баланса си. В текущата имплементация балансът на всеки играч може да бъде манипулиран чрез бек-офис приложението.

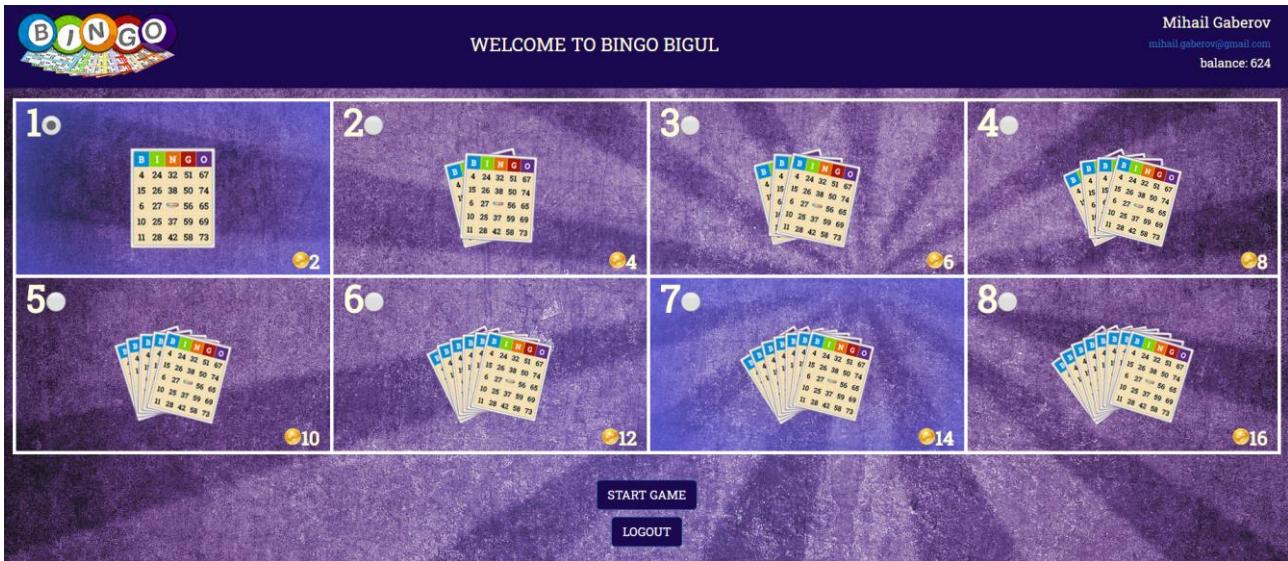


Figure 25: Лоби екран

4. Стартiranе и провеждане на една бинго игра, маркиране на изтеглени числа, спечелване на бинго – след закупуване на желан брой бинго карти, потребителят има възможност за стартиране на същинската игра. Това става с бутона „START GAME“. На играчът се представя игрови екран, на който се виждат закупените карти и се теглят случајни числа във вид на топки от сферовиден компонент. Тук играчът има възможност за маркиране на числа в картите си, в случаите когато тези числа са изтеглени от сферата. Системата използва цветови обозначения за маркираните числа, така че да става ясно когато маркираното число е наистина изтеглено или е маркирано погрешно. Изтеглени и маркирани вече числа не могат да бъдат отмаркирани.

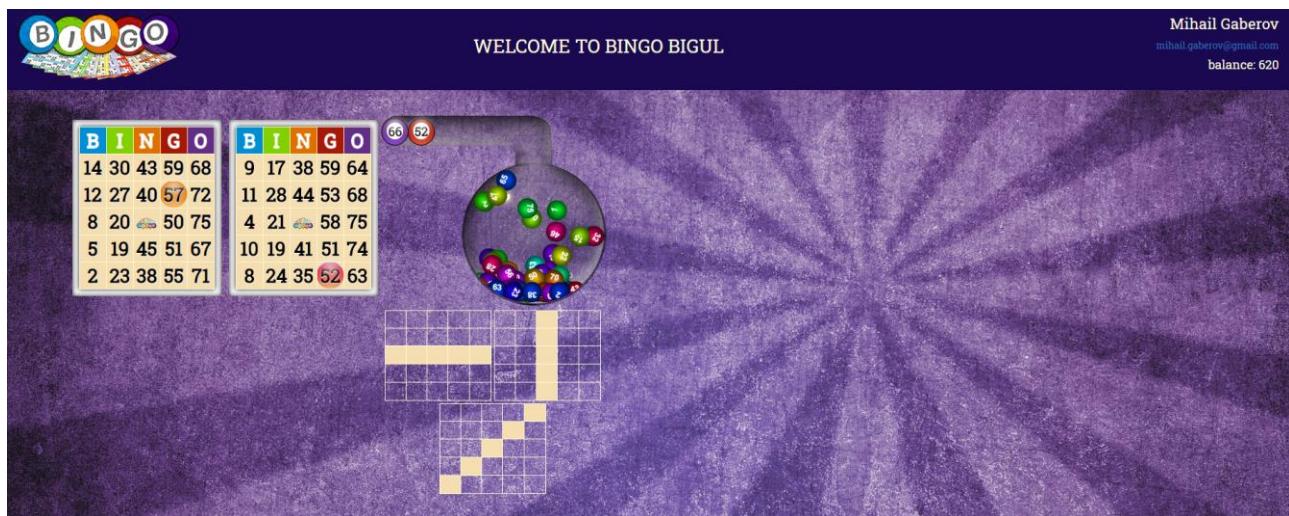


Figure 26: Игрови еcran

5. Край на бинго игра, получаване на печалби и възможност за нова игра – края на всяка игра настъпва след като са изтеглени определен брой числа. Тази стойност е конфигурируема. В края на играта, на играчът е представен диалогов прозорец, обозначаващ резултата от неговата игра. Това може да бъде брой спечелени игри и наградите за тях или неспечелена игра.

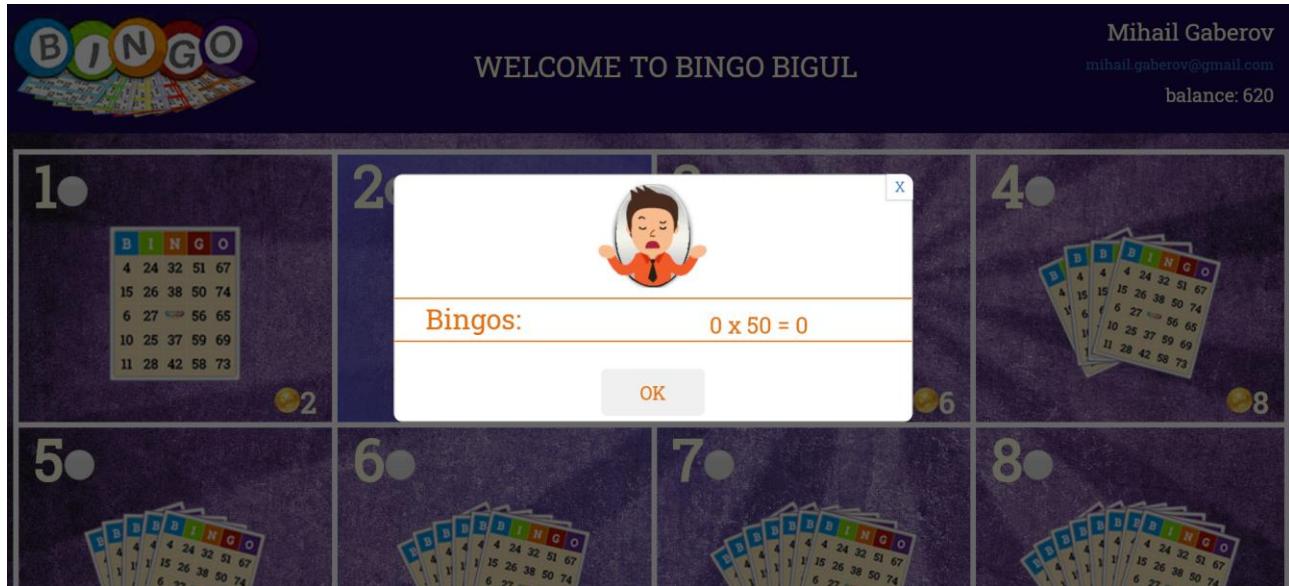


Figure 27: Край на игра

IX. Възможности за промяна

A. Разширяване възможностите на административната част

1. Въвеждане на потребителски роли – модератори, администратори и т.н.
2. Добавяне на функционалност за експортиране и импортиране на потребителските данни посредством добре познати и широко използвани технологии, например чрез употребата на .csv или .xls файлове
3. Подобряване процеса по регистрацията на нови потребители чрез въвеждане на валидация за съществуването и собствеността на въведения от тях имейл адрес – при попълване на формата за регистрация системата ще изпраща имейл със генериран линк в него, чрез който потребителя ще може да потвърди валидността на адреса и неговата собственост
4. Подобряване процеса по идентификация на вече регистрирани потребители – при коректно попълване на формата за вход, системата ще изпраща имейл съобщение, съдържащо случайно

генериран 5 цифрен код, чрез който се въвежда още едно ниво на сигурност в процеса на идентификация

5. Интеграция със система за плащания, която да позволява на играчите да правят нови депозити при изчерпване на текущия им баланс в играта

B. Развиване функционалностите на игровата част:

1. Въвеждане на gamification елементи – значки награди (badge awards), таблици с победители (leader boards)
2. Добавяне на функционалност реализираща потребителски профил, където да може да се вижда информация за играта - игровата история, дата на регистрация, спечелени бинго игри, баланс, спечелени значки и т.н.
3. Добавяне на още анимации и подобряване на съществуващите такива
4. Добавяне на други игрови елементи в самото приложение – странични малки игри, които да могат да се играят в паралел с бинго играта, например игра с карти за изтриване – Scratch Cards
5. Добавяне на функционалност реализираща т.нар. „предизвикателства“ или „мисии“ (challenges or missions) – определен тип условия, които играча трябва да спази или определени резултати, които да постигне, за да бъде награден със значка и/или парична награда за даденото предизвикателство
6. Добавяне на функционалност реализираща провеждането на Бинго турнири – игри, в които могат да участват определен брой играчи и да се състезават помежду си за постигането на определена цел, например спечелването на три поредни бинго игри
7. Въвеждане на система за разпределение на събитията в приложението – „events“ и оптимизиране на текущото състояние
8. Добавяне на функционалност реализираща озвучаване на играта – фонова музика, звуци при определени потребителски действия като кликване на бутон или маркиране на число от карта, или при определени събития като спечелване на бинго или загуба на игра
9. Добавяне на функционалност за реализиране на многоезичност на приложението – преводи на служебни текстове от ресурсни файлове или база данни
10. Оптимизиране на текущия код от гледна точка на бързодействие и последваща поддръжка – намаляване на броя DOM селектори, оптимизиране на DOM манипулациите, оптимизации за различни браузъри
11. VIP система за поощряване на най-активните играчи, т.е. да се имплементира функционалност, която автоматично да сигнализира когато например общо депозираните суми на даден играч достигнат

определенено ниво и съответно администраторите да маркират играта като ВИП, което да му носи определени бонуси, които не са налични за обикновените играчи

12. Добавяне на възможност за ръчно (от играта) определяне на времето на един игрови рунд, т.е. през колко време (секунди) да се теглят новите числа

13. Добавяне на възможност за паузиране на бинго игра – за случаите когато играчът е прекъснат и желае да продължи започнатата игра по-късно

Заключение

Текущата работа предоставя завършена платформа за реализиране и манипулиране на онлайн игра „Бинго“ от американски тип, със 75 топки. Платформата се състои от три основни части, които могат да се срещнат в почти всяко съвременно уеб приложение. Front-End или клиентска част, Back-End или сървърна част, и база данни (database). Също така платформата разполага с бек-офис приложение, чрез което могат да се манипулират данните за регистрираните играчи. При изграждане на изгледа на потребителските интерфейси в платформата е следван подход за реализиране на „отзовчив“ (responsive) уеб дизайн. Това позволява платформата да бъде лесно използвана на устройства с различен размер на екрана. В платформата са заложени съвременни практики за тестване на софтуер и планирани добавяне на още и различни по вид такива. За изграждането на приложението за администратори е използвана една от най-известните и добре развивани в днешно време библиотеки за създаване на потребителски интерфейси (A JavaScript library for building user interfaces). Това е предпоставка за улеснено бъдещо развитие и добавяне на нови функционалности за относително кратко време.

Основна цел на текущата работа е да предостави безплатен вариант за игра на всички бинго фенове или поне на тези, които харесват повече американския вариант на играта. От всичко изложено до тук може да заключим, че целта е постигната.

Когато играта бъде публикувана в интернет и бъде рекламирана по адекватен начин, тя може да се превърне в известно място във виртуалното пространство, където бинго феновете да могат да се насладят безплатно на любимата игра, както и да общуват по между си. А след като набере известна популярност, платформата има потенциал да се превърне в платен продукт и да се интегрира в игровите платформи, притежавани и развивани от големите в бранша.

От страната на разработчиците, където сме ние, можем да заключим, че дори и нищо от планираното да не се случи, удовлетворението от добре свършената работа е огромно. Огромно е и удоволствието, което носи видим и работещ резултат, какъвто е първата завършена версия на този проект!

X. Използвана литература

- <https://nodejs.org/en/>
- <https://gulpjs.com/>
- <https://www.npmjs.com/>
- <https://www.mongodb.com/>
- <https://sass-lang.com/>
- <https://blog.abelotech.com/posts/generate-random-values-nodejs-javascript/>
<https://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/>
- http://www.bgkazino.com/2013/01/blog-post_22.html
- <https://facebook.github.io/react/>
- <https://davidwalsh.name/pubsub-javascript>
- <https://scotch.io/tutorials/use-mongodb-with-a-node-application-on-heroku>
- <https://mbeaudru.github.io/modern-js-cheatsheet/>
- <https://www.hongkiat.com/blog/tools-to-coding-online/>
- <https://caniuse.com/>
- <http://www.advsys.net/ken/util/pngout.htm>
- <http://optipng.sourceforge.net/>
- <https://mediaqueri.es/>
- <http://pxtoem.com/>
- <https://type-scale.com/>
- <http://flexboxgrid.com/>
- <https://css-tricks.com/flexbox-truncated-text/>
- <https://opengameart.org/>
- http://onlineconversion.com/unix_time.htm
- <https://www.draw.io/>
- <https://regex101.com/>
- <https://css-tricks.com/viewport-sized-typography/>
- <https://www.sitepoint.com/how-to-grunt-and-gulp-your-way-to-workflow-automation/>
- <https://j11y.io/javascript/truthy-falsey/>
- <https://scotch.io/tutorials/automate-your-tasks-easily-with-gulp-js>
- <https://jskatas.org/#bundle-es6-katas>
- <https://scotch.io/bar-talk/git-cheat-sheet>
- <https://reactjs.org/tutorial/tutorial.html>
- <https://blog.risingstack.com/node-js-at-scale-understanding-node-js-event-loop/>
- <https://scotch.io/tutorials/use-mongodb-with-a-node-application-on-heroku>
- <https://daviddang.io/2015/01/12/jasmine-vs-mocha-chai-and-sinon.html>
- https://en.wikipedia.org/wiki/Behavior-driven_development
- <https://www.smashingmagazine.com/2012/06/introduction-to-javascript-unit-testing/>
- <https://www.sitepoint.com/map-reduce-functional-javascript/>
- <https://chris.beams.io/posts/git-commit/>
- <https://jrsinclair.com/articles/2016/gentle-introduction-to-javascript-tdd-intro/>
- <https://jrsinclair.com/articles/2016/tdd-should-be-fun/>

- <https://jrsinclair.com/articles/2016/one-weird-trick-that-will-change-the-way-you-code-forever-javascript-tdd/>
- <https://engineering.musefind.com/react-lifecycle-methods-how-and-when-to-use-them-2111a1b692b1>
- <https://itnext.io/6-tips-for-better-react-performance-4329d12c126b>
- <https://dev.to/jsmanifest/14-beneficial-tips-to-write-cleaner-code-in-react-apps-1gcf>

XI. Приложения

A. Каталог на схема дефинициите на колекциите в MongoDB

1. admins

```
{
  "fields": [
    {
      "name": "_id",
      "path": "_id",
      "count": 1,
      "types": [
        {
          "name": "ObjectID",
          "bsonType": "ObjectID",
          "path": "_id",
          "count": 1,
          "values": [
            "5e03162c1b02852d1c649ef4"
          ],
          "total_count": 0,
          "probability": 1,
          "unique": 1,
          "has_duplicates": false
        }
      ],
      "total_count": 1,
      "type": "ObjectID",
      "has_duplicates": false,
      "probability": 1
    },
    {
      "name": "__v",
      "path": "__v",
      "count": 1,
      "types": [
        {
          "name": "Int32",
          "bsonType": "Int32",
          "path": "__v",
          "count": 1,
          "values": [
            0
          ],
          "total_count": 1,
          "type": "Int32",
          "has_duplicates": false,
          "probability": 1
        }
      ],
      "total_count": 1,
      "type": "Int32",
      "has_duplicates": false,
      "probability": 1
    }
  ]
}
```

```

    "total_count": 0,
    "probability": 1,
    "unique": 1,
    "has_duplicates": false
  }
],
"total_count": 1,
"type": "Int32",
"has_duplicates": false,
"probability": 1
},
{
  "name": "email",
  "path": "email",
  "count": 1,
  "types": [
    {
      "name": "String",
      "bsonType": "String",
      "path": "email",
      "count": 1,
      "values": [
        "mihail.gabеров@gmail.com"
      ],
      "total_count": 0,
      "probability": 1,
      "unique": 1,
      "has_duplicates": false
    }
  ],
  "total_count": 1,
  "type": "String",
  "has_duplicates": false,
  "probability": 1
},
{
  "name": "hash",
  "path": "hash",
  "count": 1,
  "types": [
    {
      "name": "String",
      "bsonType": "String",
      "path": "hash",
      "count": 1,
      "values": [
        "4ee7998fd740d6f54b241a8ea73ee313ead62c19e8a0b7e138565ff6021757e1bd1e0046cc54370f421
b85d1a034bbfc7d1f10800e520f13cc4197445fd84f71"
      ],
      "total_count": 0,
      "probability": 1,
    }
  ]
}

```

```
"unique": 1,
  "has_duplicates": false
}
],
"total_count": 1,
"type": "String",
"has_duplicates": false,
"probability": 1
},
{
  "name": "name",
  "path": "name",
  "count": 1,
  "types": [
    {
      "name": "String",
      "bsonType": "String",
      "path": "name",
      "count": 1,
      "values": [
        "Mihail"
      ],
      "total_count": 0,
      "probability": 1,
      "unique": 1,
      "has_duplicates": false
    }
  ],
  "total_count": 1,
  "type": "String",
  "has_duplicates": false,
  "probability": 1
},
{
  "name": "salt",
  "path": "salt",
  "count": 1,
  "types": [
    {
      "name": "String",
      "bsonType": "String",
      "path": "salt",
      "count": 1,
      "values": [
        "b243ade4ea2824ec2e65f362a1085b04"
      ],
      "total_count": 0,
      "probability": 1,
      "unique": 1,
      "has_duplicates": false
    }
  ],
}
```

```

    "total_count": 1,
    "type": "String",
    "has_duplicates": false,
    "probability": 1
  }
],
"count": 1
}

```

2. users

```

{
  "fields": [
    {
      "name": "_id",
      "path": "_id",
      "count": 2,
      "types": [
        {
          "name": "ObjectID",
          "bsonType": "ObjectID",
          "path": "_id",
          "count": 2,
          "values": [
            "5c04d35f8258bd38d0f7c883",
            "5e197ac1f18f6330685ab5f0"
          ],
          "total_count": 0,
          "probability": 1,
          "unique": 2,
          "has_duplicates": false
        }
      ],
      "total_count": 2,
      "type": "ObjectID",
      "has_duplicates": false,
      "probability": 1
    },
    {
      "name": "__v",
      "path": "__v",
      "count": 2,
      "types": [
        {
          "name": "Int32",
          "bsonType": "Int32",
          "path": "__v",
          "count": 2,
          "values": [
            0,
            0
          ],
          "total_count": 0,
        }
      ]
    }
  ]
}

```

```
"probability": 1,  
"unique": 1,  
"has_duplicates": true  
}  
],  
"total_count": 2,  
"type": "Int32",  
"has_duplicates": true,  
"probability": 1  
},  
{  
"name": "balance",  
"path": "balance",  
"count": 2,  
"types": [  
{  
"name": "Int32",  
"bsonType": "Int32",  
"path": "balance",  
"count": 2,  
"values": [  
620,  
50  
],  
"total_count": 0,  
"probability": 1,  
"unique": 2,  
"has_duplicates": false  
}  
],  
"total_count": 2,  
"type": "Int32",  
"has_duplicates": false,  
"probability": 1  
},  
{  
"name": "email",  
"path": "email",  
"count": 2,  
"types": [  
{  
"name": "String",  
"bsonType": "String",  
"path": "email",  
"count": 2,  
"values": [  
"mihail.gaberov@gmail.com",  
"mihail.gaberov@zoho.com"  
],  
"total_count": 0,  
"probability": 1,  
"unique": 2,
```

```

    "has_duplicates": false
  }
],
"total_count": 2,
"type": "String",
"has_duplicates": false,
"probability": 1
},
{
  "name": "hash",
  "path": "hash",
  "count": 2,
  "types": [
    {
      "name": "String",
      "bsonType": "String",
      "path": "hash",
      "count": 2,
      "values": [
        "f7afff9d79de98d0807971502f2af69f56debbca8fdb9c18a2e1863af013db91849bbb93e5157cc4b6eb
066fc1f1d0a4da5f1a205b434258f242a5d79c22349a",
        "e8aca29fbe7836254e79c33785f6a23ee951fef88a352743800c712998a2c51112d45e8ef6d2be19661
875b1786f5bce05bfcda3b9b295f74d8330d489e5dd1a"
      ],
      "total_count": 0,
      "probability": 1,
      "unique": 2,
      "has_duplicates": false
    }
  ],
  "total_count": 2,
  "type": "String",
  "has_duplicates": false,
  "probability": 1
},
{
  "name": "name",
  "path": "name",
  "count": 2,
  "types": [
    {
      "name": "String",
      "bsonType": "String",
      "path": "name",
      "count": 2,
      "values": [
        "Mihail Gaberov",
        "Mihail Gaberov"
      ],
      "total_count": 0
    }
  ]
}

```

```
"probability": 1,  
"unique": 1,  
"has_duplicates": true  
}  
],  
"total_count": 2,  
"type": "String",  
"has_duplicates": true,  
"probability": 1  
},  
{  
"name": "salt",  
"path": "salt",  
"count": 2,  
"types": [  
{  
"name": "String",  
"bsonType": "String",  
"path": "salt",  
"count": 2,  
"values": [  
"ff2206a305402742f61ecab5270b07f4",  
"416e2ed0d9f07ac756b1562cdb839cf8"  
],  
"total_count": 0,  
"probability": 1,  
"unique": 2,  
"has_duplicates": false  

```

```

    "has_duplicates": true
  }
],
"total_count": 2,
"type": "Int32",
"has_duplicates": true,
"probability": 1
}
],
"count": 2
}

```

B. Каталог на използваните файлове

1. back-office-tests

a) Actions.test.js

```

import React from 'react';
import { cleanup, fireEvent, render, findAllByTestId, waitForElement } from
'@testing-library/react';
import Actions from '../src/back-office/components/Actions';

describe('Click some actions', () => {
  // automatically unmount and cleanup DOM after the test is finished.
  afterEach(cleanup);

  it('calls you back', async () => {
    const callback = jest.fn();

    const { container } = render(
      <Actions onAction={callback}>
    );

    const data = await waitForElement(() => findAllByTestId(container, 'actions'));

    (await data).forEach(e => fireEvent.click(e));
    const calls = callback.mock.calls;

    expect(calls.length).toEqual(3);
    expect(calls[0][0]).toEqual('info');
    expect(calls[1][0]).toEqual('edit');
    expect(calls[2][0]).toEqual('delete');
  });
});

```

b) Button.test.js

```

import React from 'react';
import ReactDOM from 'react-dom';
import TestUtils from 'react-dom/test-utils';
import Button from '../src/back-office/components/Button';

describe('Render Button components', () => {
  it('renders <a> vs <button>', () => {

```

```

const button = TestUtils.renderIntoDocument(
  <div>
    <Button>
      Hello
    </Button>
  </div>
);

expect(ReactDOM.findDOMNode(button).children[0].nodeName).toEqual('BUTTON');

const a = TestUtils.renderIntoDocument(
  <div>
    <Button href="#">
      Hello
    </Button>
  </div>
);
expect(ReactDOM.findDOMNode(a).children[0].nodeName).toEqual('A');

it('allows custom CSS classes', () => {
  const button = TestUtils.renderIntoDocument(
    <div><Button className="good bye">Hello</Button></div>
  );
  const buttonNode = ReactDOM.findDOMNode(button).children[0];
  expect(buttonNode.getAttribute('class')).toEqual('button good bye');
});

});

```

c) Dialog.test.js

```

import React from 'react';
import { cleanup, findAllByText, findByText, fireEvent, render, waitForElement } from '@testing-library/react';
import Dialog from '../../../../../src/back-office/components/Dialog';

describe('renders with action buttons', () => {
  // automatically unmount and cleanup DOM after the test is finished.
  afterEach(cleanup);

  it('can have Cancel', async () => {
    const { container } = render(<Dialog>Civilized dialog</Dialog>);

    const cancel = await waitForElement(() => findByText(container, 'Cancel'));
    expect(cancel.nodeName).toBe('SPAN');

    const ok = await waitForElement(() => findByText(container, 'OK'));
    expect(ok.textContent).toBe(Dialog.defaultProps.confirmLabel);
  });

  it('can have a single dismiss button', async () => {
    const { container } = render(<Dialog hasCancel={false} confirmLabel="Confirm">Civilized dialog</Dialog>);
    const ok = await waitForElement(() => findAllByText(container, 'Confirm'));
    expect(ok.length).toBe(1);
  });

```

```

    });

it('can be modal', () => {
  let { container } = render(<Dialog modal>Civilized dialog</Dialog>);
  expect(Array.from(document.body.classList)).toContain('dialog-modal-open');

  // removing the dialog
  container = render('');
  expect(Array.from(document.body.classList)).not.toBe('dialog-modal-open');
});

it('has head and body', async () => {
  const { container } = render(<Dialog header="Header Title">Civilized dialog</Dialog>);
  const head = await waitForElement(() => findAllByText(container, 'Header Title'));
  expect((await head).length).toBe(1);
  const body = await waitForElement(() => findAllByText(container, 'Civilized dialog'));
  expect((await body).length).toBe(1);
});

it('sends correct actions', async () => {
  const callback = jest.fn();
  let { container } = render(<Dialog onAction={callback} />);
  const cancelBtn = await waitForElement(() => findByText(container, 'Cancel'));
  fireEvent.click(cancelBtn);

  const okBtn = await waitForElement(() => findByText(container, 'OK'));
  fireEvent.click(okBtn);

  const calls = callback.mock.calls;
  expect(calls.length).toEqual(2);
  expect(calls[0][0]).toEqual('dismiss');
  expect(calls[1][0]).toEqual('confirm');
});
});

```

d) FormInput.test.js

```

jest
.dontMock('../src/back-office/components/FormInput')
.dontMock('classnames')
;

import React from 'react';
import TestUtils from 'react-dom/test-utils';

import FormInput from '../src/back-office/components/FormInput';

describe('FormInput module', () => {
  it('Renders correct input node', () => {
    expect(
      TestUtils.findRenderedDOMComponentWithTag(
        TestUtils.renderIntoDocument(<FormInput />),

```

```
'input').type).toBe('text');

expect(
  TestUtils.findRenderedDOMComponentWithTag(
    TestUtils.renderIntoDocument(<FormInput type="number"/>),
'input').type).toBe('number');

expect(
  TestUtils.findRenderedDOMComponentWithTag(
    TestUtils.renderIntoDocument(<FormInput type="text"/>),
'textarea').nodeName).toBe('TEXTAREA');

expect(
  TestUtils.findRenderedDOMComponentWithTag(
    TestUtils.renderIntoDocument(<FormInput type="email"/>),
'input').type).toBe('email');
});

it('Returns default input value', () => {
  let input = TestUtils.renderIntoDocument(<FormInput type="number"/>);
  expect(input.getValue()).toBe(String(0));
});
});
```

e) Table.test.js

```
import React from 'react';
import { cleanup, findAllByTestId, findByText, fireEvent, render, waitForElement } from '@testing-library/react';

import Table from '../../../../../src/back-office/components/Table';
import data from '../../../../../src/back-office/dummy-data';
import Store from '../../../../../src/back-office/flux/CRUDStore';

Store.init(data);

describe('Editing data', () => {
  // automatically unmount and cleanup DOM after the test is finished.
  afterEach(cleanup);

  beforeEach(() => {
    fetch.resetMocks()
  });

  // Storage Mock
  function storageMock() {
    let storage = {};

    return {
     setItem: function (key, value) {
        storage[key] = value || '';
      },
     getItem: function (key) {
        return storage[key] || null;
      },
     removeItem: function (key) {
        delete storage[key];
      },
     get length() {

```

```

        return Object.keys(storage).length;
    },
    key: function (i) {
        let keys = Object.keys(storage);
        return keys[i] || null;
    }
};

it('Saves new data', async () => {
    const window = document.defaultView;
    window.localStorage = storageMock();
    const { container } = render(<Table />);
    const newName = 'Gosho';

    const cell = await waitForElement(() => findByText(container, 'Mihail Gaberov'));
    fireEvent.doubleClick(cell);

    cell.getElementsByTagName('input')[0].value = newName;
    fireEvent.submit(cell.getElementsByTagName('form')[0]);
    expect(cell.textContent).toBe(newName);
});

it('Deletes data', async () => {
    fetch.mockResponseOnce(JSON.stringify({ data: 'ok' }));

    const { container } = render(<Table />);
    const deleteIcon = await waitForElement(() => findByText(container, 'x'));
    fireEvent.click(deleteIcon);

    const confirmationDialogOkBtn = await waitForElement(() => findByText(container, 'Delete'));
    fireEvent.click(confirmationDialogOkBtn);

    const cells = await waitForElement(() => findAllById(container, 'td'));
    expect(cells.length).toBe(4);
});

});

```

2. bingo-api

a) config/passport.js

```

const passport = require('passport');
const LocalStrategy = require('passport-local').Strategy;
const mongoose = require('mongoose');
const User = mongoose.model('Users');
const Admin = mongoose.model('Admins');

passport.use('userLogin', new LocalStrategy({
    usernameField: 'email'
},
function(username, password, done) {
    User.findOne({ email: username }, function (err, user) {
        if (err) { return done(err); }

```

```

if (!user) {
  return done(null, false, {
    message: 'User not found'
  });
}

if (!user.validPassword(password)) {
  return done(null, false, {
    message: 'Password is wrong'
  });
}

return done(null, user);
});

passport.use('adminLogin', new LocalStrategy({
  usernameField: 'email'
},
function(username, password, done) {

Admin.findOne({ email: username }, function (err, admin) {
  if (err) { return done(err); }

  if (!admin) {
    return done(null, false, {
      message: 'Admin not found'
    });
  }

  if (!admin.validPassword(password)) {
    return done(null, false, {
      message: 'Password is wrong'
    });
  }

  return done(null, admin);
});
});
));

```

b) controllers/admins-auth.js

```

const passport = require('passport');
const mongoose = require('mongoose');
const Admin = mongoose.model('Admins');

module.exports.register = function (req, res) {
  const admin = new Admin();

  admin.name = req.body.name;
  admin.email = req.body.email;

  // Check for existence
  Admin.count({email: admin.email}, function (err, count) {
    if (count > 0) {
      res.status(200);
      res.json({
        message: 'Email already exists'
      });
    } else {
      admin.save(function (err) {
        if (err) {
          res.status(500);
          res.json({
            message: 'Internal Server Error'
          });
        } else {
          res.status(201);
          res.json({
            message: 'Admin created successfully'
          });
        }
      });
    }
  });
};

```

```

    'isExisted': true
  });
} else {
  admin.setPassword(req.body.password);
  admin.save(function (err) {
    const token = admin.generateJwt();
    res.status(200);
    res.json({
      'token': token
    });
  });
}
};

module.exports.loginAdmin = function (req, res) {
  passport.authenticate('adminLogin', function (err, admin, info) {
    let token;

    if (err) {
      res.status(404).json(err);
      return;
    }

    if (admin) {
      token = admin.generateJwt();
      res.status(200);
      res.json({
        'token': token
      });
    } else {
      res.status(401).json(info);
    }
  })(req, res);
};

```

c) controllers/back-office.js

```

const mongoose = require('mongoose');
const User = mongoose.model('Users');

module.exports.getPlayersData = function (req, res) {
  User
    .find()
    .exec(function (err, allUsers) {
      res.status(200).json(allUsers);
    });
};

module.exports.createPlayer = function (req, res) {
  var user = new User();

  user.name = req.body.objPlayerData.name;
  user.email = req.body.objPlayerData.email;

  // Check for existence
  User.count({email: user.email}, function (err, count) {
    if (count > 0) {
      res.status(200);
    }
  });
};

```

```

res.json({
  'isExisted': true
});
} else {
  user.setBalance(req.body.objPlayerData.balance);
  user.setWins(req.body.objPlayerData.wins);
  user.setPassword(req.body.objPlayerData.password);
  user.save(function (err) {
    var token = user.generateJwt();
    res.status(200);
    res.json({
      'token': token
    });
  });
}
};

module.exports.deletePlayer = function (req, res) {
  if (!req.body.email) {
    res.status(401).json({
      "message": "Non existing user"
    });
  } else {
    User
      .remove({ "email": req.body.email })
      .exec(function (err, data) {
        res.status(200).json(data);
      });
  }
};

module.exports.updatePlayerData = function (req, res) {
  if (!req.body.objPlayerData.email) {
    res.status(401).json({
      "message": "UnauthorizedError: unauthorized attempt to update player data"
    });
  } else {
    User.findOne({ email: req.body.objPlayerData.email }, function (err, user) {
      if (req.body.objPlayerData) {
        user.name = req.body.objPlayerData.name;
        user.email = req.body.objPlayerData.email;
        user.setBalance(req.body.objPlayerData.balance);
        user.setWins(req.body.objPlayerData.wins);

        if (req.body.objPlayerData.password)
          user.setPassword(req.body.objPlayerData.password);

      }
      user.save(function (err, user) {
        res.status(200).json(user);
      });
    });
  }
};

```

d) controllers/profile.js

```
const mongoose = require('mongoose');
const User = mongoose.model('Users');

module.exports.profileRead = function (req, res) {
  // If no user ID exists in the JWT return a 401
  if (!req.payload._id) {
    res.status(401).json({
      "message": "UnauthorizedError: private profile"
    });
  } else {
    // Otherwise continue
    User
      .findById(req.payload._id)
      .exec(function (err, user) {
        res.status(200).json(user);
      });
  }
};

module.exports.setNewBalance = function (req, res) {
  if (!req.payload._id) {
    res.status(401).json({
      "message": "UnauthorizedError: unauthorized attempt to set balance"
    });
  } else {
    User.findOne({email: req.body.email}, function (err, user) {
      if (req.body.spending) {
        user.setBalance(user.balance - req.body.newSum);
      } else {
        user.setBalance(user.balance + req.body.newSum);
      }

      user.save(function (err, user) {
        res.status(200).json(user);
      });
    });
  }
};

module.exports.setWins = function (req, res) {
  if (!req.payload._id) {
    res.status(401).json({
      "message": "UnauthorizedError: unauthorized attempt to set wins"
    });
  } else {
    User.findOne({email: req.body.email}, function (err, user) {
      user.setWins(user.wins + req.body.wins);

      user.save(function (err, user) {
        res.status(200).json(user);
      });
    });
  }
};
```

e) controllers/users-auth.js

```
const passport = require('passport');
const mongoose = require('mongoose');
const User = mongoose.model('Users');

module.exports.register = function (req, res) {
  const user = new User();

  user.name = req.body.name;
  user.email = req.body.email;

  // Check for existence
  User.count({ email: user.email }, function (err, count) {
    if (count > 0) {
      res.status(200);
      res.json({
        'isExisted': true,
      });
    } else {
      user.setBalance(50);
      user.setWins(0);
      user.setPassword(req.body.password);
      user.save(function (err) {
        const token = user.generateJwt();
        res.status(200);
        res.json({
          'token': token,
        });
      });
    }
  });
};

module.exports.login = function (req, res) {

  passport.authenticate('userLogin', function (err, user, info) {
    if (err) {
      res.status(404).json(err);
      return;
    }

    if (user) {
      res.status(200);
      res.json({
        'token': user.generateJwt()
      });
    } else {
      res.status(401).json(info);
    }
  })(req, res);
};
```

f) models/admins.js

```
const mongoose = require('mongoose');
const crypto = require('crypto');
const jwt = require('jsonwebtoken');
require('dotenv').config();
```

```

const adminSchema = new mongoose.Schema({
  email: {
    type: String,
    unique: true,
    required: true
  },
  name: {
    type: String,
    required: true
  },
  hash: String,
  salt: String
});

adminSchema.methods.setPassword = function(password) {
  this.salt = crypto.randomBytes(16).toString('hex');
  this.hash = crypto.pbkdf2Sync(password, this.salt, 1000, 64,
  'sha512').toString('hex');
};

adminSchema.methods.validPassword = function(password) {
  const hash = crypto.pbkdf2Sync(password, this.salt, 1000, 64,
  'sha512').toString('hex');
  return this.hash === hash;
};

adminSchema.methods.generateJwt = function() {
  const expiry = new Date();
  expiry.setDate(expiry.getDate() + 7);

  return jwt.sign({
    _id: this._id,
    email: this.email,
    name: this.name,
    exp: parseInt(expiry.getTime() / 1000)
  }, process.env.DB_SECRET);
};

mongoose.model('Admins', adminSchema);

```

g) models/db.js

```

const mongoose = require('mongoose');
let gracefulShutdown;
const db = mongoose.connection;

mongoose
  .connect(process.env.DB_URI, {
    useUnifiedTopology: true,
    useNewUrlParser: true,
  })
  .then(() => console.log('DB Connected.'))
  .catch(err => {
    console.log(`DB Connection Error: ${err.message}`);
  });
mongoose.set('useCreateIndex', true);

db.on('connected', function () {

```

```

    console.log('Mongoose connected to ' + process.env.DB_URI);
});
db.on('error', function (err) {
  console.log('Mongoose connection error: ' + err);
});
db.on('disconnected', function () {
  console.log('Mongoose disconnected');
});

gracefulShutdown = function (msg, callback) {
  db.close(function () {
    console.log('Mongoose disconnected through ' + msg);
    callback();
  });
};

process.once('SIGUSR2', function () {
  gracefulShutdown('nodemon restart', function () {
    process.kill(process.pid, 'SIGUSR2');
  });
});

process.on('SIGINT', function () {
  gracefulShutdown('app termination', function () {
    process.exit(0);
  });
});

require('./users');
require('./admins');

```

h) models/users.js

```

const mongoose = require('mongoose');
const crypto = require('crypto');
const jwt = require('jsonwebtoken');

const userSchema = new mongoose.Schema({
  email: {
    type: String,
    unique: true,
    required: true
  },
  name: {
    type: String,
    required: true
  },
  hash: String,
  salt: String,
  balance: {
    type: Number,
    required: true
  },
  wins: {
    type: Number,
    required: true
  }
});

```

```

userSchema.methods.setBalance = function(amount) {
  this.balance = amount;
};

userSchema.methods.setWins = function(count) {
  this.wins = count;
};

userSchema.methods.setPassword = function(password) {
  this.salt = crypto.randomBytes(16).toString('hex');
  this.hash = crypto.pbkdf2Sync(password, this.salt, 1000, 64,
    'sha512').toString('hex');
};

userSchema.methods.validPassword = function(password) {
  const hash = crypto.pbkdf2Sync(password, this.salt, 1000, 64,
    'sha512').toString('hex');
  return this.hash === hash;
};

userSchema.methods.generateJwt = function() {
  const expiry = new Date();
  expiry.setDate(expiry.getDate() + 7);

  return jwt.sign({
    _id: this._id,
    email: this.email,
    name: this.name,
    balance: this.balance,
    wins: this.wins,
    exp: parseInt(expiry.getTime() / 1000)
  }, process.env.DB_SECRET);
};

mongoose.model('Users', userSchema);

```

i) routes/index.js

```

const express = require('express');
const router = express.Router();
const jwt = require('express-jwt');
require('dotenv').config();

const auth = jwt({
  secret: process.env.DB_SECRET,
  userProperty: 'payload'
});

// Back office app
const ctrlBackOffice = require('../controllers/back-office');
const ctrlAdminsAuth = require('../controllers/admins-auth');
router.get('/getPlayersData', auth, ctrlBackOffice.getPlayersData);
router.post('/createPlayer', auth, ctrlBackOffice.createPlayer);
router.post('/deletePlayer', auth, ctrlBackOffice.deletePlayer);
router.post('/updatePlayerData', auth, ctrlBackOffice.updatePlayerData);
router.post('/loginAdmin', ctrlAdminsAuth.loginAdmin);
// For the time being the admin roles will be added only manually
router.post('/registerAdmin', ctrlAdminsAuth.register);

```

```
// Main app
const ctrlProfile = require('../controllers/profile');
const ctrlAuth = require('../controllers/users-auth');
router.post('/setNewBalance', auth, ctrlProfile.setNewBalance);
router.get('/profile', auth, ctrlProfile.profileRead);
router.post('/register', ctrlAuth.register);
router.post('/login', ctrlAuth.login);
router.post('/setWins', auth, ctrlProfile.setWins);

module.exports = router;
```

3. front-end-tests

a) api/api-controller.test.js

```
import ApiController from '../../src/API/api-controller';

describe('Api Controller', () => {
  function storageMock() {
    const storage = {};

    return {
     setItem: function(key, value) {
        storage[key] = value || '';
      },
     getItem: function(key) {
        return storage[key] || null;
      },
     removeItem: function(key) {
        delete storage[key];
      },
     getLength() {
        return Object.keys(storage).length;
      },
     key: function(i) {
        const keys = Object.keys(storage);
        return keys[i] || null;
      }
    };
  }

  test('Should initialize View Ctrl to interact with the views', () => {
    const window = document.defaultView;
    window.localStorage = storageMock();

    const apiCtrl = new ApiController();
    expect(apiCtrl.viewCtrl).not.toBe(undefined);
  });
});
```

b) blower/blower.test.js

```
import Blower from '../../src/blower/blower';

describe('Blower module', () => {
  test('Should create a blower component', (done) => {
    let el = document.createElement('canvas');
    const blower = new Blower(el);
```

```

expect(blower).not.toBe(undefined);
done();
});

test('Should init with stopped animation', (done) => {
  let el = document.createElement('canvas');
  let blower = new Blower(el);
  expect(blower.init).toEqual(expect.objectContaining({ play: false,
isPlaying: false }));
  done();
});

test('Should create an array with 75 ball objects', (done) => {
  let el = document.createElement('canvas');
  let blower = new Blower(el);
  expect(blower.balls.length).toEqual(75);
  done();
});

test('Should have mechanism for starting the animation', (done) => {
  let el = document.createElement('canvas');
  let blower = new Blower(el);
  expect(typeof blower.startAnimation).toBe('function');
  done();
});

test('Should have mechanism for stopping the animation', (done) => {
  let el = document.createElement('canvas');
  let blower = new Blower(el);
  expect(typeof blower.stopAnimation).toBe('function');
  done();
});
}

```

c) card/card.test.js

```

import Card from '../../src/card/card';

describe('Card object', () => {
const objCard = {
  col1: [ 10, 14, 13, 15, 5 ],
  col2: [ 27, 23, 21, 29, 22 ],
  col3: [ 39, 37, 'x', 32, 33 ],
  col4: [ 56, 51, 60, 57, 59 ],
  col5: [ 72, 74, 63, 71, 70 ]
};

test('Should create new HTML card', () => {
  const card = new Card(objCard);
  expect(card).toBeTruthy();
});

test('Should provide method for marking numbers', () => {
  expect(Card.markNumber).toBeTruthy();
});

test('Should provide method for marking number cells as drawn', () => {
  expect(Card.markDrawnNumber).toBeTruthy();
});
}

```

```
test('Should create new HTML card that has table cells with appropriate IDs', () => {
    const card = new Card(objCard);

expect(card.children[0].firstElementChild.children[1].children[0].id).toEqual('11');

expect(card.children[0].firstElementChild.children[1].children[1].id).toEqual('21');

expect(card.children[0].firstElementChild.children[1].children[2].id).toEqual('31');

expect(card.children[0].firstElementChild.children[1].children[3].id).toEqual('41');

expect(card.children[0].firstElementChild.children[1].children[4].id).toEqual('51');

expect(card.children[0].firstElementChild.children[2].children[0].id).toEqual('12');

expect(card.children[0].firstElementChild.children[2].children[1].id).toEqual('22');

expect(card.children[0].firstElementChild.children[2].children[2].id).toEqual('32');

expect(card.children[0].firstElementChild.children[2].children[3].id).toEqual('42');

expect(card.children[0].firstElementChild.children[2].children[4].id).toEqual('52');

expect(card.children[0].firstElementChild.children[3].children[0].id).toEqual('13');

expect(card.children[0].firstElementChild.children[3].children[1].id).toEqual('23');

expect(card.children[0].firstElementChild.children[3].children[2].id).toEqual('33');

expect(card.children[0].firstElementChild.children[3].children[3].id).toEqual('43');

expect(card.children[0].firstElementChild.children[3].children[4].id).toEqual('53');

expect(card.children[0].firstElementChild.children[4].children[0].id).toEqual('14');

expect(card.children[0].firstElementChild.children[4].children[1].id).toEqual('24');
```

```

expect(card.children[0].firstElementChild.children[4].children[2].id).toEqual('34');

expect(card.children[0].firstElementChild.children[4].children[3].id).toEqual('44');

expect(card.children[0].firstElementChild.children[4].children[4].id).toEqual('54');

expect(card.children[0].firstElementChild.children[5].children[0].id).toEqual('15');

expect(card.children[0].firstElementChild.children[5].children[1].id).toEqual('25');

expect(card.children[0].firstElementChild.children[5].children[2].id).toEqual('35');

expect(card.children[0].firstElementChild.children[5].children[3].id).toEqual('45');

expect(card.children[0].firstElementChild.children[5].children[4].id).toEqual('55');
    }
);
};


```

d) card/card-drawer.test.js

```

import CardDrawer from '../../../../../src/card/card-drawer';

describe('Card Drawer', () => {
  const objCard = {
    col1: [ 10, 14, 13, 15, 5 ],
    col2: [ 27, 23, 21, 29, 22 ],
    col3: [ 39, 37, 'x', 32, 33 ],
    col4: [ 56, 51, 60, 57, 59 ],
    col5: [ 72, 74, 63, 71, 70 ]
  };

  test('Should get the number of the cards to be generated', () => {
    const el = document.createElement('div');
    let toBeGenerated = CardDrawer.draw({ 'card1': objCard }, el);
    expect(toBeGenerated.length).toEqual(1);
  });

  test(
    'Should create a div element with id "card" with a Bingo card table inside',
    () => {
      const htmlCard = CardDrawer.generateCardTable(objCard);

      expect(htmlCard.id).toEqual('card');
      expect(htmlCard.children[0].tagName).toEqual('TABLE');

      expect(htmlCard.children[0].firstElementChild.children.length).toEqual(6);
    }
  );
}


```

```

/* row 1 */

expect(htmlCard.children[0].firstElementChild.children[1].children.length).toEqual(5);
    // col 1

expect(parseInt(htmlCard.children[0].firstElementChild.children[1].children[0]
    .innerHTML)).toEqual(10);
    // col 2

expect(parseInt(htmlCard.children[0].firstElementChild.children[1].children[1]
    .innerHTML)).toEqual(27);
    // col 3

expect(parseInt(htmlCard.children[0].firstElementChild.children[1].children[2]
    .innerHTML)).toEqual(39);
    // col 4

expect(parseInt(htmlCard.children[0].firstElementChild.children[1].children[3]
    .innerHTML)).toEqual(56);
    // col 5

expect(parseInt(htmlCard.children[0].firstElementChild.children[1].children[4]
    .innerHTML)).toEqual(72);

/* row 2 */

expect(htmlCard.children[0].firstElementChild.children[2].children.length).toEqual(5);
    // col 1

expect(parseInt(htmlCard.children[0].firstElementChild.children[2].children[0]
    .innerHTML)).toEqual(14);
    // col 2

expect(parseInt(htmlCard.children[0].firstElementChild.children[2].children[1]
    .innerHTML)).toEqual(23);
    // col 3

expect(parseInt(htmlCard.children[0].firstElementChild.children[2].children[2]
    .innerHTML)).toEqual(37);
    // col 4

expect(parseInt(htmlCard.children[0].firstElementChild.children[2].children[3]
    .innerHTML)).toEqual(51);
    // col 5

expect(parseInt(htmlCard.children[0].firstElementChild.children[2].children[4]
    .innerHTML)).toEqual(74);

/* row 3 */

expect(htmlCard.children[0].firstElementChild.children[3].children.length).toEqual(5);
    // col 1

expect(parseInt(htmlCard.children[0].firstElementChild.children[3].children[0]
    .innerHTML)).toEqual(13);
    // col 2

```

```

expect(parseInt(htmlCard.children[0].firstElementChild.children[3].children[1]
    .innerHTML)).toEqual(21);
    // col 3

expect(htmlCard.children[0].firstElementChild.children[3].children[2].innerHTML)
    .toEqual('x');
    // col 4

expect(parseInt(htmlCard.children[0].firstElementChild.children[3].children[3]
    .innerHTML)).toEqual(60);
    // col 5

expect(parseInt(htmlCard.children[0].firstElementChild.children[3].children[4]
    .innerHTML)).toEqual(63);

    /* row 4 */

expect(htmlCard.children[0].firstElementChild.children[4].children.length).toEqual(5);
    // col 1

expect(parseInt(htmlCard.children[0].firstElementChild.children[4].children[0]
    .innerHTML)).toEqual(15);
    // col 2

expect(parseInt(htmlCard.children[0].firstElementChild.children[4].children[1]
    .innerHTML)).toEqual(29);
    // col 3

expect(parseInt(htmlCard.children[0].firstElementChild.children[4].children[2]
    .innerHTML)).toEqual(32);
    // col 4

expect(parseInt(htmlCard.children[0].firstElementChild.children[4].children[3]
    .innerHTML)).toEqual(57);
    // col 5

expect(parseInt(htmlCard.children[0].firstElementChild.children[4].children[4]
    .innerHTML)).toEqual(71);

    /* row 5 */

expect(htmlCard.children[0].firstElementChild.children[5].children.length).toEqual(5);
    // col 1

expect(parseInt(htmlCard.children[0].firstElementChild.children[5].children[0]
    .innerHTML)).toEqual(5);
    // col 2

expect(parseInt(htmlCard.children[0].firstElementChild.children[5].children[1]
    .innerHTML)).toEqual(22);
    // col 3

expect(parseInt(htmlCard.children[0].firstElementChild.children[5].children[2]
    .innerHTML)).toEqual(33);
    // col 4

expect(parseInt(htmlCard.children[0].firstElementChild.children[5].children[3]
    .innerHTML));

```

```

    .innerHTML)).toEqual(59);
    // col 5

expect(parseInt(htmlCard.children[0].firstElementChild.children[5].children[4]
    .innerHTML)).toEqual(70);
}
);

});

```

e) card/card-generator.test.js

```

import CardGenerator from '../../src/card/card-generator';

describe('Card Generator', () => {

const arrAmericanNumbers = [
  1, 2, 3, 4, 5,
  6, 7, 8, 9, 10,
  11, 12, 13, 14, 15,
  16, 17, 18, 19, 20,
  21, 22, 23, 24, 25,
  26, 27, 28, 29, 30,
  31, 32, 33, 34, 35,
  36, 37, 38, 39, 40,
  41, 42, 43, 44, 45,
  46, 47, 48, 49, 50,
  51, 52, 53, 54, 55,
  56, 57, 58, 59, 60,
  61, 62, 63, 64, 65,
  66, 67, 68, 69, 70,
  71, 72, 73, 74, 75
];

const cardGen = new CardGenerator({gameConf: {numbers: arrAmericanNumbers}});

test('Should initialize Card creation services', () => {
  expect(cardGen).toBeTruthy();
});

test('Should generate a given number of cards', () => {
  let count = 4;

  const cards = cardGen.generateCards(count);

  expect(cards).toBeTruthy();

  while (count > 0) {
    expect(cards).toHaveProperty('card' + count);
    count--;
  }
});

```

f) card/card-numbers-generator.test.js

```
import GenerateCardNumbers from '../../src/card/card-numbers-generator';

describe('Card Numbers Generator', () => {

  let arrAmericanNumbers = [1, 2, 3, 4, 5,
    6, 7, 8, 9, 10,
    11, 12, 13, 14, 15,
    16, 17, 18, 19, 20,
    21, 22, 23, 24, 25,
    26, 27, 28, 29, 30,
    31, 32, 33, 34, 35,
    36, 37, 38, 39, 40,
    41, 42, 43, 44, 45,
    46, 47, 48, 49, 50,
    51, 52, 53, 54, 55,
    56, 57, 58, 59, 60,
    61, 62, 63, 64, 65,
    66, 67, 68, 69, 70,
    71, 72, 73, 74, 75
  ];

  let cardGen = new GenerateCardNumbers({ 'gameConf': { 'numbers': arrAmericanNumbers } });

  test('Should initialize with an array of 75 numbers', () => {
    expect(cardGen.arrAmericanNumbers.length).toEqual(75);
  });

  test('Should generate a random card with 24 numbers divided by columns', () => {
    const card = cardGen.generate();

    expect(card).toBeTruthy();
    expect(card).toHaveProperty('col1');
    expect(card['col1'].length).toEqual(5);
    expect(card).toHaveProperty('col2');
    expect(card['col2'].length).toEqual(5);
    expect(card).toHaveProperty('col3');
    expect(card['col3'].length).toEqual(5);
    expect(card).toHaveProperty('col4');
    expect(card['col4'].length).toEqual(5);
    expect(card).toHaveProperty('col5');
    expect(card['col5'].length).toEqual(5);
  });
});
```

g) dauber/ball.test.js

```
import Ball from '../../src/dauber/ball';
import PubSub from '../../src/events/pubsub-service';

jest.useFakeTimers();

describe('Ball module', () => {
```

```

test('Should create a ball object', (done) => {
  const pb = new PubSub();
  const ball = new Ball(34, pb, 'original');
  expect(ball).toBeTruthy();
  done();
});

test('Should create a ball object with given number', (done) => {
  const pb = new PubSub();
  const ball = new Ball(34, pb, 'original');
  expect(ball.elNumber.innerText).toBeTruthy();
  expect(ball.elNumber.innerText).toEqual(34);
  done();
});

test('Should create a ball object with given css class', (done) => {
  const pb = new PubSub();
  const ball = new Ball(34, pb, { name: 'original' });
  expect(ball.elBall.className).toEqual('original_ballN');
  done();
});

test('Should be able to draw itself', () => {
  const pb = new PubSub();
  const ball = new Ball(34, pb, 'original');
  expect(ball.draw).toBeTruthy();
  const el = document.createElement('div');
  el.setAttribute('id', '#tube');
  ball.draw(el, 4, true);
  expect(setTimeout).toHaveBeenCalledWithTimes(1);
});

test('Should have methods moveVerticalHorizontal and animate', (done) => {
  const pb = new PubSub();
  const ball = new Ball(34, pb, 'original');
  expect(ball.move).toBeTruthy();
  expect(ball.animate).toBeTruthy();
  done();
});

```

h) dauber/dauber.test.js

```

import { Utils } from '../../../../../src/utils/utils';
import Dauber from '../../../../../src/dauber/dauber';

describe('Dauber module', () => {
  let conf = {
    "gameConf": {
      "id": "1",
      "name": "American Bingo",
      "numbers": [
        1, 2, 3, 4, 5,
        6, 7, 8, 9, 10,
        11, 12, 13, 14, 15,
        16, 17, 18, 19, 20,
        21, 22, 23, 24, 25,
        26, 27, 28, 29, 30,
      ]
    }
  };

```

```

    31, 32, 33, 34, 35,
    36, 37, 38, 39, 40,
    41, 42, 43, 44, 45,
    46, 47, 48, 49, 50,
    51, 52, 53, 54, 55,
    56, 57, 58, 59, 60,
    61, 62, 63, 64, 65,
    66, 67, 68, 69, 70,
    71, 72, 73, 74, 75
  ]
}

};

test('Should create a dauber module', (done) => {
  const selector = document.createElement('section');
  const dauber = new Dauber(conf, selector);
  expect(dauber).toBeTruthy();
  done();
});

test('Should be able to start the drawing of numbers/balls', (done) => {
  const selector = document.createElement('section');
  const dauber = new Dauber(conf, selector);
  expect(dauber.startDrawing).toBeTruthy();
  done();
});

test('Should be able to stop the drawing of numbers/balls', (done) => {
  const selector = document.createElement('section');
  const dauber = new Dauber(conf, selector);
  expect(dauber.endGame).toBeTruthy();
  done();
});

test('Should produce each number from the range only once', (done) => {
  const selector = document.createElement('section');
  const dauber = new Dauber(conf, selector);
  expect(dauber.drawNewNumber).toBeTruthy();
  let arrDrawnNumbers = [];

  for (let i = 0, l = conf.gameConf.numbers.length; i < l; ++i) {
    let num = dauber.drawNewNumber();
    if (num !== undefined)
      arrDrawnNumbers.push(num);
  }

  const arrRes = Utils.eliminateDuplicates(arrDrawnNumbers);
  expect(arrDrawnNumbers.length).toEqual(arrRes.length);
  done();
});

test(
  'Should be able to moveVerticalHorizontal the balls when 5 are visible and hide the first one',
  (done) => {
    const selector = document.createElement('div');
    document.body.appendChild(selector);

    const dauber = new Dauber(conf, selector);
    expect(dauber.arrVisibleBalls).toBeTruthy();
  }
);

```

```

    const divEl = document.createElement('div');
    dauber.arrVisibleBalls = [
      { elBall: divEl },
      { elBall: divEl },
      { elBall: divEl },
      { elBall: divEl },
      { elBall: divEl }
    ];
    dauber.animateVisibleBalls();
    expect(dauber.arrVisibleBalls.length).toEqual(4);
    expect(dauber.arrVisibleBalls[0].elBall.style.display).toEqual('none');
    done();
  }
);
}
);

```

i) events/events-consts.test.js

```

import { EventsConsts } from '../../../../../src/events/events-consts';

describe('Bingo Events Constants', () => {
  test('Should contain LOGOUT constant', () => {
    expect(EventsConsts.LOGOUT).toEqual('logout');
  });

  test('Should contain NEW_BALL_DRAWN constant', () => {
    expect(EventsConsts.NEW_BALL_DRAWN).toEqual('newBallDrawn');
  });

  test('Should contain START_GAME constant', () => {
    expect(EventsConsts.START_GAME).toEqual('startGame');
  });

  test('Should contain END_GAME constant', () => {
    expect(EventsConsts.END_GAME).toEqual('endGame');
  });

  test('Should contain BINGO constant', () => {
    expect(EventsConsts.BINGO).toEqual('bingo');
  });

  test('Should contain PRIZE_WON constant', () => {
    expect(EventsConsts.PRIZE_WON).toEqual('prizeWon');
  });

  test('Should contain FLYING_PRIZE_ANIMATION_ENDS constant', () => {
    expect(EventsConsts.FLYING_PRIZE_ANIMATION_ENDS).toEqual('flyingPrizeAnimationEnds');
  });

  test('Should contain ENOUGH_BALANCE constant', () => {
    expect(EventsConsts.ENOUGH_BALANCE).toEqual('enoughBalance');
  });

  test('Should contain NOT_ENOUGH_BALANCE constant', () => {
    expect(EventsConsts.NOT_ENOUGH_BALANCE).toEqual('notEnoughBalance');
  });
});

```

```
});  
});
```

j) events/pubsub-service.test.js

```
import PubSubService from '../../src/events/pubsub-service';

describe('PubSub Service', () => {
  const pubsub = new PubSubService();

  test('Should initialize topics object', () => {
    expect(pubsub.topics).toBeDefined();
    expect(typeof pubsub.topics).toBe('object');
  });

  test('Should initialize hOP variable', () => {
    expect(pubsub.hOP).toBeDefined();
  });

  test('Should publish events with topic and info', () => {
    pubsub.topics['test'] = [jest.fn()];
    pubsub.publish('test', {});
    expect(pubsub.topics['test']).toBeDefined();
    expect(pubsub.topics['test'][0]).toHaveBeenCalledTimes(1);
  });

  test('Should subscribe to events with topic and attach a listener', () => {
    const listener = jest.fn();
    pubsub.subscribe('eventName', listener);
    expect(pubsub.topics['eventName']).toBeDefined();
    pubsub.publish('eventName', {});
    expect(listener).toHaveBeenCalledTimes(1);
  });

  test('Should remove subscriptions', () => {
    const removeMe = () => {};
    pubsub.subscribe('removeMeEvent', removeMe);
    expect(pubsub.topics['removeMeEvent']).toBeDefined();
    pubsub.remove('removeMeEvent');
    expect(pubsub.topics['removeMeEvent'][0]).not.toEqual(removeMe);
  });
});
```

k) initializer/initializer.test.js

```
import Initializer from '../../src/initializer/initializer';

describe('App Initializer', () => {
  const conf = {
    "gameConf": {
      "id": "1",
      "name": "American Bingo",
      "numbers": [
        1, 2, 3, 4, 5,
        6, 7, 8, 9, 10,
        11, 12, 13, 14, 15,
        16, 17, 18, 19, 20,
      ]
    }
  };
});
```

```

21, 22, 23, 24, 25,
26, 27, 28, 29, 30,
31, 32, 33, 34, 35,
36, 37, 38, 39, 40,
41, 42, 43, 44, 45,
46, 47, 48, 49, 50,
51, 52, 53, 54, 55,
56, 57, 58, 59, 60,
61, 62, 63, 64, 65,
66, 67, 68, 69, 70,
71, 72, 73, 74, 75
],
"skin": {
  "name": "original"
},
"appTitle": "Welcome To Bingo Bigul",
"turnsCount": 4,
"freeSpotImgPath": "<img src='../../../../images/small_logo_30x30.png' alt=' />",
"drawIntervalSeconds": 3,
"beforeStartGameSeconds": 3,
"marketCards": true,
"dauber": true,
"playingCards": true,
"mainGame": true,
"winningDialog": true,
"cardPrice": 2,
"winPatternsAnimModule": true
}
};

test('Should have method for applying the game configurations', () => {
  expect(Initializer.hasOwnProperty('applyConfigurations')).toBeTruthy();
});

test('Should have method for setting the app title', () => {
  expect(Initializer.hasOwnProperty('setTitle')).toBeTruthy();
});

test('Should have method for adding Winning Dialog', () => {
  expect(Initializer.hasOwnProperty('addWinningDialog')).toBeTruthy();
});

test('Should have method for setting card prices', () => {
  expect(Initializer.hasOwnProperty('setCardPrices')).toBeTruthy();
});

test('Should have method for adding Winning Animation module', () => {
expect(Initializer.hasOwnProperty('addWinPatternAnimModule')).toBeTruthy();
});

test('Should have method for adding Dauber module', () => {
  expect(Initializer.hasOwnProperty('addDauber')).toBeTruthy();
});

test('Should have method for adding Logout button', () => {
  expect(Initializer.hasOwnProperty('addLogoutBtn')).toBeTruthy();
});

```

```

test('Should set the app meta title', () => {
  const el = document.createElement('title');
  el.innerText = 'test';
  document.head.appendChild(el);
  Initializer.setTitle('Welcome');
  expect(el.innerText).toEqual('Welcome');
});

test('Should add Winning Dialog instance', () => {
  let wd = Initializer.addWinningDialog(true);
  expect(wd).toBeDefined();
  wd = Initializer.addWinningDialog(false);
  expect(wd).not.toBeDefined();
});

test('Should set card prices', () => {
  let cards = document.createElement('div');
  cards.setAttribute('class', 'cards');
  let price = document.createElement('div');
  price.setAttribute('class', 'price');
  let radioInput = document.createElement('input');
  radioInput.setAttribute('type', 'radio');
  radioInput.checked = true;
  radioInput.setAttribute('value', 7);
  cards.appendChild(price);
  cards.appendChild(radioInput);
  document.body.appendChild(cards);
  const arrElements = Initializer.setCardPrices(conf.gameConf.cardPrice,
  cards);

expect(arrElements[0].querySelector('.price').innerHTML.endsWith('14')).toBeTruthy();
});

test('Should add Winning Pattern Animations module', () => {
  const elWinPatternsAnimModule = document.createElement('div');
  elWinPatternsAnimModule.setAttribute('id', 'winPatternsAnimModule');
  const horPattern = document.createElement('div');
  horPattern.setAttribute('id', 'horizontal');
  const verPattern = document.createElement('div');
  verPattern.setAttribute('id', 'vertical');
  const diagPattern = document.createElement('div');
  diagPattern.setAttribute('id', 'diagonal');
  elWinPatternsAnimModule.appendChild(horPattern);
  elWinPatternsAnimModule.appendChild(verPattern);
  elWinPatternsAnimModule.appendChild(diagPattern);
  document.body.appendChild(elWinPatternsAnimModule);
  let res = Initializer.addWinPatternAnimModule(true);
  expect(res).toBeDefined();
  res = Initializer.addWinPatternAnimModule(false);
  expect(res).not.toBeDefined();
});

test('Should add start game button', () => {
  let startBtn = Initializer.addStartButton(conf,
  Initializer.addMarketPlace(true));
  expect(startBtn).toBeDefined();
  conf.gameConf.mainGame = false;
  startBtn = Initializer.addStartButton(conf,
  Initializer.addMarketPlace(true));
}

```

```

expect(startBtn).not.toBeDefined();
});

test('Should create a Timer object', () => {
  const timer = Initializer.getTimer(conf);
  expect(timer).toBeDefined();
  expect(typeof timer).toBe('object');
  expect(timer.hasOwnProperty('element')).toBeTruthy();
  expect(timer.hasOwnProperty('seconds')).toBeTruthy();
  expect(timer.hasOwnProperty('eventName')).toBeTruthy();
  expect(timer.hasOwnProperty('isVisible')).toBeTruthy();
});

test('Should add dauber elements', () => {
  const tube = document.createElement('div');
  tube.setAttribute('id', 'tube');
  const blowerBaloon = document.createElement('canvas');
  blowerBaloon.setAttribute('id', 'blower-balloon');
  document.body.appendChild(tube);
  document.body.appendChild(blowerBaloon);
  let res = Initializer.addDauber(conf);
  expect(res.dauber).toBeDefined();
  expect(res.blower).toBeDefined();
});
});

```

I) market-place/market-cards.test.js

```

import MarketCards from '../../../../../src/market-place/market-cards';

describe('Market Cards module', () => {
  test('Should have container - html element', () => {
    const container = document.createElement('div');
    const marketCards = new MarketCards(container);
    expect(marketCards.container).toBeDefined();
    expect(marketCards.container.tagName).toEqual('DIV');
  });

  test('Should define the count of the purchased cards', () => {
    const arrRadioButtons = [
      {
        type: 'radio',
        checked: true,
        value: 4
      },
      {
        type: 'radio',
        checked: false,
        value: 5
      }
    ];

    const countCards = MarketCards.getPurchasedCardsCount(arrRadioButtons);
    expect(countCards).toEqual(4);
  });

  test(
    'Should set the correct price depending on how many cards are offered',
    () => {

```

```

        let cards = document.createElement('div');
        cards.innerHTML = '1 <input type="radio" id="one" name="marketCards" value="1" checked="checked">' +
            '' +
            '<div class="price"></div>';

        MarketCards.setCardPrices(5, [cards]);
        expect(cards.querySelector('.price').innerHTML).toEqual('<i class="price-icon"></i>5');
    }
};


```

m) utils/animator.test.js

```

import Animator from '../../../../../src/utils/animator';

describe('Animator module', () => {
    test('Should provide linear method used in animations', () => {
        expect(Animator.linear).toBeDefined();
    });

    test('Should provide quadratic method used in animations', () => {
        expect(Animator.quad).toBeDefined();
    });

    test('Should provide bounce method used in animations', () => {
        expect(Animator.bounce).toBeDefined();
    });

    test('Should provide animate method used in animations', () => {
        expect(Animator.animate).toBeDefined();
    });

    test('Should provide moveVerticalHorizontal method used in animations', () => {
        expect(Animator.moveVerticalHorizontal).toBeDefined();
    });

    test(
        'Should animate moving an element from its current to a given coordinates',
        (done) => {
            const el = document.createElement('div');
            el.style.top = '5px';
            el.style.left = '5px';

            Animator.animate = jest.fn();

            Animator.moveVerticalHorizontal(el, 10, 10, Animator.linear, 200,
                'px');

            setTimeout(() => {
                expect(el.style.top).toEqual('10px');
                expect(el.style.left).toEqual('10px');
            }, 500);

            expect(Animator.animate).toHaveBeenCalledTimes(2);
        }
    );
});


```

```

        done();
    }

);

test('Should animate rotating of a given element', () => {
  const el = document.createElement('div');
  el.style.transform = 0;

  Animator.rotateElement(el, 30, Animator.linear, 300);
  setTimeout(() => {
    expect(el.style.transform).toEqual('rotate(30deg)');
  }, 500);
});

test(
  'Should have method for triggering an event after given duration expires',
  () => {
    let caught = false;
    document.addEventListener('test', () => {
      caught = true;
    });

    Animator.dispatchEventAfterDuration('test', 1100);

    setTimeout(() => {
      expect(caught).toBeTruthy();
    }, 1100);
  }
);
}
);

```

n) utils/bangup.test.js

```

import Bangup from '../../src/utils/bangup';

describe('Bangup module', () => {
  test(
    'Should have the necessary params - container, start sum, end sum and duration',
    () => {
      const window = document.defaultView;
      window.requestAnimationFrame = ()=> {};
      const elContainer = document.createElement('div');
      const bangup = new Bangup(elContainer, 0, 100, 3);
      expect(bangup.container).toBeDefined();
      expect(bangup.startTime).toBeDefined();
      expect(bangup.fromSum).toBeDefined();
      expect(bangup.toSum).toBeDefined();
      expect(bangup.duration).toBeDefined();
    }
  );

  test('Should reach the end sum after the duration has expired', () => {
    const window = document.defaultView;
    window.requestAnimationFrame = ()=> {};
    const duration = 3;
    const elContainer = document.createElement('div');
    new Bangup(elContainer, 0, 100, duration);
  });
}
);

```

```

    setTimeout(() => {
      expect(elContainer.innerHTML).toEqual('100');
    }, duration * 1000);
  );
}

```

o) utils/number-generator.test.js

```

import { NumbersGenerator } from '../../src/utils/numbers-generator';

describe('Numbers generation util', () => {

  test('Should produce random number between min and max values', () => {
    let min = 1, max = 5, res1 = 0, res2 = 0;
    while (res1 === res2) {
      res1 = NumbersGenerator.getRandomNumber(min, max);
      res2 = NumbersGenerator.getRandomNumber(min, max);
    }
    expect(res1).not.toEqual(res2);
  });

  test(
    'Should produce 5 different random numbers out of array of 15 numbers',
    () => {
      const arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15];

      const arrRes = NumbersGenerator.getColumnNumbers(arr);

      expect(arrRes.length).toEqual(5);
      expect(arrRes[0]).not.toEqual(arrRes[1]);
      expect(arrRes[1]).not.toEqual(arrRes[2]);
      expect(arrRes[2]).not.toEqual(arrRes[3]);
      expect(arrRes[3]).not.toEqual(arrRes[4]);
    }
  );

  test(
    'Should do random generation the first column, 5 numbers in range: B is 1-15',
    () => {
      const arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15];

      const arrRes = NumbersGenerator.getColumnNumbers(arr, 5);

      expect(arrRes.length).toEqual(5);

      for (let i = 0; i < arrRes.length; i++) {
        expect(arrRes[i]).not.toBeLessThan(1);
        expect(arrRes[i]).not.toBeGreaterThanOrEqual(15);
      }
    }
  );

  test(
    'Should do random generation the second column, 5 numbers in range: I is 16-30',
    () => {
      const arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15];
      const arrRes = NumbersGenerator.getColumnNumbers(arr, 5);
      expect(arrRes.length).toEqual(5);

      for (let i = 0; i < arrRes.length; i++) {
        expect(arrRes[i]).not.toBeLessThan(16);
        expect(arrRes[i]).not.toBeGreaterThanOrEqual(30);
      }
    }
  );
}

```

```

() => {
  const arr = [16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
30];

  const arrRes = NumbersGenerator.getColumnNumbers(arr, 5);

  expect(arrRes.length).toEqual(5);

  for (let i = 0; i < arrRes.length; i++) {
    expect(arrRes[i]).not.toBeLessThan(16);
    expect(arrRes[i]).not.toBeGreaterThan(30);
  }
};

test(
  'Should do random generation the third column, 5 numbers in range: N is
31-45',
  () => {
    const arr = [31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44,
45];

    const arrRes = NumbersGenerator.getColumnNumbers(arr, 5);

    expect(arrRes.length).toEqual(5);

    for (let i = 0; i < arrRes.length; i++) {
      expect(arrRes[i]).not.toBeLessThan(31);
      expect(arrRes[i]).not.toBeGreaterThan(45);
    }
  }
);

test(
  'Should do random generation the fourth column, 5 numbers in range: G is
46-60',
  () => {
    const arr = [46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
60];

    const arrRes = NumbersGenerator.getColumnNumbers(arr, 5);

    expect(arrRes.length).toEqual(5);

    for (let i = 0; i < arrRes.length; i++) {
      expect(arrRes[i]).not.toBeLessThan(46);
      expect(arrRes[i]).not.toBeGreaterThan(60);
    }
  }
);

test(
  'Should do random generation the fifth column, 5 numbers in range: O is
61-75',
  () => {
    const arr = [61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74,
75];

    const arrRes = NumbersGenerator.getColumnNumbers(arr, 5);

```

```

expect(arrRes.length).toEqual(5);

for (let i = 0; i < arrRes.length; i++) {
  expect(arrRes[i]).not.toBeLessThan(61);
  expect(arrRes[i]).not.toBeGreaterThan(75);
}

};

};

}

```

p) utils/timer.test.js

```

import Timer from '../../../../../src/utils/timer';
import { EventsConsts } from '../../../../../src/events/events-consts';

describe('Timer module', () => {
  test('Should create new Timer object', () => {
    const timer = new Timer(document.querySelector('#timerContainer'), 5,
EventsConsts.START_GAME, true);
    expect(timer).toBeDefined();
  });

  test('Should accept 3 parameters – seconds, eventName and isVisible', () =>
{
  const timer = new Timer(document.querySelector('#timerContainer'), 5,
EventsConsts.START_GAME, true);
  expect(timer.hasOwnProperty('seconds')).toBeTruthy();
  expect(timer.hasOwnProperty('eventName')).toBeTruthy();
  expect(timer.hasOwnProperty('isVisible')).toBeTruthy();
});

  test('Should accept 3 parameters types – number, string and boolean', () =>
{
  const timer = new Timer(document.querySelector('#timerContainer'), 5,
EventsConsts.START_GAME, true);
  expect(timer.seconds).toEqual(5);
  expect(timer.eventName).toContain(EventsConsts.START_GAME);
  expect(timer.isVisible).toBeTruthy();
});

  test('Should provide method for pulsating', () => {
    const timer = new Timer(document.querySelector('#timerContainer'), 5,
EventsConsts.START_GAME, true);
    expect(timer.startCounting).toBeDefined();
  });

  test('Should animate pulse for given time and the to be hidden', () => {
    const el = document.createElement('div');
    const timer = new Timer(el, 5, EventsConsts.START_GAME, true);
    timer.startCounting();
    setTimeout(() => {
      expect(timer.element.style.display).toEqual('none');
    }, (timer.seconds) * 1000);
  });

  test('Should provide method hiding the timer', () => {
    const timer = new Timer(document.querySelector('#timerContainer'), 5,
EventsConsts.START_GAME, true);
    expect(timer.hide).toBeDefined();
  });
}

```

```

    });

test('Should hide the Timer container whit the relevant method', () => {
  const el = document.createElement('div');
  const timer = new Timer(el, 5, EventsConsts.START_GAME, true);
  timer.hide();
  expect(timer.element.style.display).toEqual('none');
});
}

```

q) utils/utils.test.js

```

import { Utils } from '../../../../../src/utils/utils';

describe('Utils service', () => {
  test('Should provide a method for eliminating duplicates in array', (done) => {
    const arr = [1,2,3,4,5,5];
    const arrRes = Utils.eliminateDuplicates(arr);
    expect(arrRes.length).toEqual(5);
    done();
  });

  test('Should provide a method for defining the ball css class', (done) => {
    const num = 15;
    const resColor = Utils.getCssClassByNumber(num);
    expect(resColor).toEqual('ballB');
    done();
  });

  test(
    'Should provide a method for counting a given element in array',
    (done) => {
      const arr = [1, 2, 3, 1];
      const len = Utils.countInArray(arr, 1);
      expect(len).toEqual(2);
      done();
    }
  );
}

```

r) utils/view-manipulator.test.js

```

import ViewManipulator from '../../../../../src/utils/view-manipulator';
import PubSubService from '../../../../../src/events/pubsub-service';

describe('ViewManipulator module', () => {
  test('Should select the necessary dom elements', (done) => {
    const pubsub = new PubSubService();
    const viewMan = new ViewManipulator(pubsub);
    expect(viewMan.signInLink).toBeDefined();
    expect(viewMan.registerForm).toBeDefined();
    expect(viewMan.loginForm).toBeDefined();
    expect(viewMan.marketPlace).toBeDefined();
    expect(viewMan.btnCloseAlertMsg).toBeDefined();
    expect(viewMan.isOnLoginPage).toBeDefined();
    expect(viewMan.isOnLoginPage).toBeTruthy();
  });
}

```

```

done();
});

test('Should attach the necessary dom events listeners', () => {
  const pubsub = new PubSubService();
  const viewMan = new ViewManipulator(pubsub);
  const spy = spyOn(viewMan, 'attachViewListeners');
  setTimeout(() => {
    expect(spy).toHaveBeenCalled();
  });
});

test('Should be subscribed to the LOGOUT pubsub event', () => {
  const pubsub = new PubSubService();
  let viewManipulator = new ViewManipulator(pubsub);
  const spy = spyOn(pubsub, 'subscribe');
  setTimeout(() => {
    expect(spy).toHaveBeenCalled();
  });
});

test('Should provide a method for manipulating alert messages', (done) => {
  const elAlertMsg = document.createElement('div');
  const elMessageText = document.createElement('div');
  elMessageText.setAttribute('id', 'messageText');
  elAlertMsg.appendChild(elMessageText);
  ViewManipulator.toggleErrorMessageView(elAlertMsg, 'test text', true);

  expect(elAlertMsg.style.display).toEqual('block');
  expect(elMessageText.innerText).toEqual('test text');

  ViewManipulator.toggleErrorMessageView(elAlertMsg, 'test text', false);

  expect(elAlertMsg.style.display).toEqual('none');
  done();
});

test('Should provide a method for showing/hiding a given element', (done) => {
  const el = document.createElement('div');
  el.setAttribute('id', 'dauber');
  document.body.appendChild(el);
  ViewManipulator.toggleVisibility(el, false);
  expect(el.style.display).toEqual('none');
  done();
});

test('Should provide a method for updating the view/screen state', (done) => {
  const elGameWrapper = document.createElement('div');
  const elRegisterPage = document.createElement('div');
  let isLoggedIn = false;

  ViewManipulator.updateViewState(elGameWrapper, elRegisterPage, isLoggedIn);
  expect(elGameWrapper.style.display).toEqual('none');
  expect(elRegisterPage.style.display).toEqual('block');

  isLoggedIn = true;
});

```

```

    ViewManipulator.updateViewState(elGameWrapper, elRegisterPage,
isloggedIn);
expect(elGameWrapper.style.display).toEqual('block');
expect(elRegisterPage.style.display).toEqual('none');
done();
});
});

```

s) utils/winning-patterns.test.js

```

import { WinningPatterns } from '../../../../../src/utils/winning-patterns';

describe('WinningPatterns module', () => {

test('Should provide a method for checking horizontal Bingo pattern', () => {
let arr = ['11', '21', '31', '41', '51'];
let res = WinningPatterns.checkHorizontalPattern(arr);
expect(res).toBeTruthy();

arr = ['11', '22', '31', '41', '51'];
res = WinningPatterns.checkHorizontalPattern(arr);
expect(res).not.toBeTruthy();

arr = ['13', '23', '43', '53'];
res = WinningPatterns.checkHorizontalPattern(arr);
expect(res).toBeTruthy();

arr = ['13', '23', '44', '53'];
res = WinningPatterns.checkHorizontalPattern(arr);
expect(res).not.toBeTruthy();
});

test('Should provide a method for checking vertical Bingo pattern', () => {
let arr = ['11', '12', '13', '14', '15'];
let res = WinningPatterns.checkVerticalPattern(arr);
expect(res).toBeTruthy();

arr = ['11', '12', '13', '14', '22'];
res = WinningPatterns.checkVerticalPattern(arr);
expect(res).not.toBeTruthy();

arr = ['31', '32', '34', '35'];
res = WinningPatterns.checkVerticalPattern(arr);
expect(res).toBeTruthy();

arr = ['31', '32', '34', '43'];
res = WinningPatterns.checkVerticalPattern(arr);
expect(res).not.toBeTruthy();
});

test('Should provide a method for checking diagonal Bingo pattern', () => {
let arr = ["11", "22", "44", "55"];
let res = WinningPatterns.checkDiagonalPattern(arr);
expect(res).toBeTruthy();

arr = ["11", "22", "44", "53"];
res = WinningPatterns.checkDiagonalPattern(arr);
expect(res).not.toBeTruthy();
});

```

```

arr = ["15", "24", "42", "51"];
res = WinningPatterns.checkDiagonalPattern(arr);
expect(res).toBeTruthy();

arr = ["15", "24", "42", "25"];
res = WinningPatterns.checkDiagonalPattern(arr);
expect(res).not.toBeTruthy();
});

test('Should provide a method for checking corners Bingo pattern', () => {
let arr = ["11", "15", "51", "55"];
let res = WinningPatterns.checkCornersPattern(arr);
expect(res).toBeTruthy();

arr = ["11", "15", "51", "44"];
res = WinningPatterns.checkCornersPattern(arr);
expect(res).not.toBeTruthy();
});
});

```

t) winning/flying-prize.test.js

```

import FlyingPrize from '../../../../../src/winning/flying-prize';
import Animator from '../../../../../src/utils/animator'

describe('FlyingPrize module', () => {
test('Should contain the prize sum to animate flying', () => {
const fp = new FlyingPrize(123);
expect(fp.sum).toEqual(123);
});

test('Should contain a method for animating the prize', () => {
expect(FlyingPrize.animatePrizeFlying).toBeDefined();
});

test('Should use Animator method moveDiagonally', () => {
Animator.moveDiagonally = jest.fn();
FlyingPrize.animatePrizeFlying(123);
expect(Animator.moveDiagonally).toHaveBeenCalled();
});
});

```

u) winning/win-patterns-anim-module.test.js

```

import WinPatternsAnimModule from '../../../../../src/winning/win-patterns-anim-
module';

describe('WinPatternsAnimModule module', () => {
test('Should get element container, rows, cols and patterns', () => {
const container = document.createElement('section');
let wpam = new WinPatternsAnimModule(container, 5, 5, 'horizontal');
expect(wpam.elem).toBeDefined();
expect(wpam.rows).toBeDefined();
expect(wpam.cols).toBeDefined();
expect(wpam.pattern).toBeDefined();
});

test(

```

```

'Should call createElement to create the DOM structure of the module',
() => {
  const container = document.createElement('section');
  const wpam = new WinPatternsAnimModule(container, 5, 5, 'horizontal');
  wpam.createElement = jest.fn();
  setTimeout(() => expect(wpam.createElement).toHaveBeenCalled());
}
);

test(
  'Should call startAnimation method to animate the winning patterns',
  () => {
    const container = document.createElement('section');
    const wpam = new WinPatternsAnimModule(container, 5, 5, 'horizontal');
    wpam.startAnimation = jest.fn();
    setTimeout(() => expect(wpam.startAnimation).toHaveBeenCalled());
  }
);

test('Should be able to clear the animation module table', () => {
  const container = document.createElement('section');
  const wpam = new WinPatternsAnimModule(container, 5, 5, 'horizontal');
  expect(wpam.clearTable).toBeDefined();
});

test('Should have method for animating horizontal pattern', () => {
  const container = document.createElement('section');
  const wpam = new WinPatternsAnimModule(container, 5, 5, 'horizontal');
  expect(wpam.startHorizontalAnim).toBeDefined();
});

test('Should have method for animating vertical pattern', () => {
  const container = document.createElement('section');
  const wpam = new WinPatternsAnimModule(container, 5, 5, 'vertical');
  expect(wpam.startVerticalAnim).toBeDefined();
});

test('Should have method for animating diagonal pattern', () => {
  const container = document.createElement('section');
  const wpam = new WinPatternsAnimModule(container, 5, 5, 'diagonal');
  expect(wpam.startDiagonalAnim).toBeDefined();
});

```

v) winning/winning-dialog.test.js

```

import WinningDialog from '../../src/winning/winning-dialog';

describe('WinningDialog module', () => {
  test('Should get the ID of a DOM element to contain the modal', () => {
    const wd = new WinningDialog('#id');
    expect(wd.elementID).toEqual('#id');
  });

  test('Should attach the necessary listeners', () => {
    const spy = spyOn(WinningDialog, 'attachListeners');
    new WinningDialog('#id');
    expect(spy).toHaveBeenCalled();
  });
}
);

```

```

    });

test('Should assign the appropriate css class depending on how many bingos are won',
() => {
  let className = WinningDialog.getHeaderImgClass(1);
  expect(className).toEqual('winner-one-bingo');
  className = WinningDialog.getHeaderImgClass(0);
  expect(className).toEqual('no-bingo');
}
);
}
);

```

4. src

a) API/api-consts.js

```

const ApiConsts = {
  CONF: 'http://localhost:8000/config.json',
  REGISTER: 'http://localhost:8888/bingo-api/register',
  LOGIN: 'http://localhost:8888/bingo-api/login',
  PROFILE: 'http://localhost:8888/bingo-api/profile',
  SET_BALANCE: 'http://localhost:8888/bingo-api/setNewBalance',
  GET_PLAYERS_DATA: 'http://localhost:8888/bingo-api/getPlayersData',
  CREATE_PLAYER: 'http://localhost:8888/bingo-api/createPlayer',
  UPDATE_PLAYER_DATA: 'http://localhost:8888/bingo-api/updatePlayerData',
  DELETE_PLAYER: 'http://localhost:8888/bingo-api/deletePlayer',
  LOGIN_ADMIN: 'http://localhost:8888/bingo-api/loginAdmin',
  SET_WINS: 'http://localhost:8888/bingo-api/setWins'
};

export default ApiConsts;

```

b) API/api-controller.js

```

import { EventsConsts } from '../events/events-consts';
import PubSubService from '../events/pubsu-sub-service';
import LocalStorageService from '../local-storage/local-storage-service';
import DbService from './db-service';
import ViewManipulator from '../utils/view-manipulator';

class ApiController {
  constructor() {
    if (LocalStorageService.isLoggedIn()) {
      ViewManipulator.updateViewState(undefined, undefined,
        LocalStorageService.isLoggedIn());
    }
  }

  this.pubsub = new PubSubService();
  this.viewCtrl = new ViewManipulator(this.pubsub);
}

logout() {
  LocalStorageService.logout();
  this.pubsub.publish(EventsConsts.LOGOUT, {
    isLogout: true
  });
}

```

```

}

static login() {
  const elEmail = document.querySelector('#email');
  const elPass = document.querySelector('#password');

  if (elEmail === null || elEmail.value === undefined ||
    elPass === null || elPass.value === undefined) {
    console.log('Not valid user data.');
    return;
  }

  const promiseLogin = DbService.loginPlayer(elEmail.value, elPass.value);
  promiseLogin.then((val) => {
    if (val.token) {
      ViewManipulator.updateViewState(undefined, undefined, true);
      LocalStorageService.saveToken(val.token);
      ViewManipulator.showUserInfo();
    } else {
      ViewManipulator.toggleErrorMessageView(document.querySelector('#alertMsg'),
        'Wrong login details.', true);
    }
  });
}

static register() {
  const elName = document.querySelector('#registerName');
  const elEmail = document.querySelector('#registerEmail');
  const elPass = document.querySelector('#registerPassword');

  if (elName === null || elName.value === undefined ||
    elEmail === null || elEmail.value === undefined ||
    elPass === null || elPass.value === undefined) {
    console.log('Not valid user data.');
    return;
  }

  const promiseReg = DbService.registerPlayer(elName.value, elEmail.value,
    elPass.value);
  promiseReg.then((val) => {
    if (val) {
      if (val.isExisted) {

        ViewManipulator.toggleErrorMessageView(document.querySelector('#alertMsg'),
          'User already existed.', true);
      } else {
        LocalStorageService.saveToken(val.token);
        ViewManipulator.updateViewState(undefined, undefined,
          LocalStorageService.isLoggedIn());
        ViewManipulator.showUserInfo();
      }
    } else {
      console.log('Show error message for registration failed.');
    }
  });
}

static getProfileInfo() {
  return LocalStorageService.currentUser();
}

```

```

}

static isLoggedIn() {
  return LocalStorageService.isLoggedIn();
}

static setNewBalance(sum, isSpending = true) {
  const promiseSetNewBalance =
DbService.updateBalance(ApiController.getProfileInfo().email, sum,
isSpending);

promiseSetNewBalance.then((val) => {
  if (val) {
    ViewManipulator.updateBalance(val.balance - sum, val.balance);
  } else {
    console.log('Show error message for setting new balance failed.');
  }
});
}

static getPlayerBalancePromise() {
  return DbService.getPlayerBalance().then((val) => {
    LocalStorageService.saveBalance(val);
    return val;
  });
}

static getPlayerBalanceFromStorage() {
  return LocalStorageService.getBalance();
}

static getPlayersDataPromise() {
  return DbService.getAllPlayersData().then((val) => {
    return val;
  });
}

static createPlayerPromise(objPlayerData) {
  return DbService.createPlayer(objPlayerData).then((val) => {
    return val;
  });
}

static deletePlayerPromise(playerEmail) {
  return DbService.deletePlayer(playerEmail).then((val) => {
    return val;
  });
}

static updatePlayerDataPromise(objPlayerData) {
  return DbService.updatePlayerData(objPlayerData).then((val) => {
    return val;
  });
}

static loginAdminPromise(objCredentials) {
  return DbService.loginAdmin(objCredentials).then((val) => {
    return val;
  });
}

```

```

static setBingoWins(wins) {
  const setWinsPromises =
DbService.setWins(ApiController.getProfileInfo().email, wins);

  setWinsPromises.then((val) => {
    if (val) {
      console.log('bingos won: ', wins);
    } else {
      console.log('Setting new wins count failed.');
    }
  });
}

export default ApiController;

```

c) API/db-service.js

```

import LocalStorageService from '../local-storage/local-storage-service';
import ApiConsts from './api-consts';

class DbService {
  static loginPlayer(email, pass) {
    return fetch(ApiConsts.LOGIN, {
      method: 'POST',
      body: JSON.stringify({
        email: email,
        password: pass
      }),
      mode: 'cors',
      redirect: 'follow',
      headers: new Headers({
        'Content-Type': 'application/json'
      })
    }).then((res) => {
      return res.json();
    }).then((returnedValue) => {
      return returnedValue;
    }).catch(function (err) {
      console.log('>>> Fetching error on login: ', err);
    });
  }

  static registerPlayer(name, email, pass) {
    return fetch(ApiConsts.REGISTER, {
      method: 'POST',
      body: JSON.stringify({
        name: name,
        email: email,
        password: pass
      }),
      mode: 'cors',
      redirect: 'follow',
      headers: new Headers({
        'Content-Type': 'application/json'
      })
    }).then((res) => {
      return res.json();
    });
  }
}

```

```

}).then((returnedValue) => {
  return returnedValue;
}).catch(function (err) {
  console.log('>>> Fetching error on register: ', err);
});
}

static getPlayerBalance() {
  return fetch(ApiConsts.PROFILE, {
    method: 'GET',
    mode: 'cors',
    redirect: 'follow',
    headers: {
      Authorization: 'Bearer ' + LocalStorageService.getToken()
    }
  }).then((res) => {
    return res.json();
  }).then((returnedValue) => {
    return returnedValue.balance;
  }).catch(function (err) {
    console.log('>>> Fetching error on getting player balance: ', err);
  });
}

static updateBalance(email, sum, spending) {
  return fetch(ApiConsts.SET_BALANCE, {
    method: 'POST',
    body: JSON.stringify({
      email: email,
      newSum: sum,
      spending: spending
    }),
    mode: 'cors',
    redirect: 'follow',
    headers: new Headers({
      'Content-Type': 'application/json',
      Authorization: 'Bearer ' + LocalStorageService.getToken()
    })
  }).then((res) => {
    return res.json();
  }).then((returnedValue) => {
    return returnedValue;
  }).catch(function (err) {
    console.log('>>> Fetching error on updating balance: ', err);
  });
}

static getAllPlayersData() {
  return fetch(ApiConsts.GET_PLAYERS_DATA, {
    method: 'GET',
    mode: 'cors',
    redirect: 'follow',
    headers: new Headers({
      'Content-Type': 'application/json',
      Authorization: 'Bearer ' + LocalStorageService.getAdminToken()
    })
  }).then((res) => {
    return res.json();
  }).then((returnedValue) => {
    return returnedValue;
  });
}

```

```

}).catch(function (err) {
  console.log('>>> Fetching error on getting all players data: ', err);
});

static createPlayer(objPlayerData) {
  return fetch(ApiConsts.CREATE_PLAYER, {
    method: 'POST',
    body: JSON.stringify({
      objPlayerData: objPlayerData
    }),
    mode: 'cors',
    redirect: 'follow',
    headers: new Headers({
      'Content-Type': 'application/json',
      Authorization: 'Bearer ' + LocalStorageService.getAdminToken()
    })
  }).then((res) => {
    return res.json();
  }).then((returnedValue) => {
    return returnedValue;
  }).catch(function (err) {
    console.log('>>> Fetching error on creating player: ', err);
  });
}

static deletePlayer(playerEmail) {
  return fetch(ApiConsts.DELETE_PLAYER, {
    method: 'POST',
    body: JSON.stringify({
      email: playerEmail
    }),
    mode: 'cors',
    redirect: 'follow',
    headers: new Headers({
      'Content-Type': 'application/json',
      Authorization: 'Bearer ' + LocalStorageService.getAdminToken()
    })
  }).then((res) => {
    return res.json();
  }).then((returnedValue) => {
    return returnedValue;
  }).catch(function (err) {
    console.log('>>> Fetching error on deleting player: ', err);
  });
}

static updatePlayerData(objPlayerData) {
  return fetch(ApiConsts.UPDATE_PLAYER_DATA, {
    method: 'POST',
    body: JSON.stringify({
      objPlayerData: objPlayerData
    }),
    mode: 'cors',
    redirect: 'follow',
    headers: new Headers({
      'Content-Type': 'application/json',
      Authorization: 'Bearer ' + LocalStorageService.getAdminToken()
    })
  }).then((res) => {
}

```

```

    return res.json();
}).then((returnedValue) => {
  return returnedValue;
}).catch(function (err) {
  console.log('>>> Fetching error on updating player data: ', err);
});
}

static loginAdmin(objCredentials) {
  return fetch(ApiConsts.LOGIN_ADMIN, {
    method: 'POST',
    body: JSON.stringify({
      email: objCredentials.email,
      password: objCredentials.password
    }),
    mode: 'cors',
    redirect: 'follow',
    headers: new Headers({
      'Content-Type': 'application/json'
    })
  }).then((res) => {
    return res.json();
  }).then((returnedValue) => {
    return returnedValue;
  }).catch(function (err) {
    console.log('>>> Fetching error on admin login: ', err);
  });
}

static setWins(email, wins) {
  return fetch(ApiConsts.SET_WINS, {
    method: 'POST',
    body: JSON.stringify({
      email: email,
      wins: wins
    }),
    mode: 'cors',
    redirect: 'follow',
    headers: new Headers({
      'Content-Type': 'application/json',
      Authorization: 'Bearer ' + LocalStorageService.getToken()
    })
  }).then((res) => {
    return res.json();
  }).then((returnedValue) => {
    return returnedValue;
  }).catch(function (err) {
    console.log('>>> Fetching error on setting wins: ', err);
  });
}

export default DbService;

```

d) back-office/components/Actions.js

```

import React from 'react';
const Actions = props =>

```

```

<div id="actions">
  <span
    data-testid="actions"
    tabIndex="0"
    className="actions-info"
    title="More info"
    onClick={props.onAction.bind(null, 'info')}>&#8505;</span>
  <span
    data-testid="actions"
    tabIndex="0"
    className="actions-edit"
    title="Edit"
    onClick={props.onAction.bind(null, 'edit')}>&#10000;</span>
  <span
    data-testid="actions"
    tabIndex="0"
    className="actions-delete"
    title="Delete"
    onClick={props.onAction.bind(null, 'delete')}>x</span>
</div>

export default Actions;

```

e) back-office/components/BackOffice.js

```

/* @flow */

import Button from './Button';
import CRUDActions from '../flux/CRUDActions';
import CRUDStore from '../flux/CRUDStore';
import Dialog from './Dialog';
import Table from './Table';
import Form from './Form';
import React, {Component} from 'react';
import EventsConsts from './EventsConsts';

type State = {
  addNew: boolean,
  count: number,
  errorUserExists: boolean,
  errorDeletion: boolean,
  errorUpdating: boolean,
  isLoggedIn: boolean
};

class BackOffice extends Component {

  state: State;

  constructor() {
    super();
    this.state = {
      addNew: false,
      errorUserExists: false,
      errorDeletion: false,
      errorUpdating: false,
      count: CRUDStore.getCount(),
      isLoggedIn: CRUDStore.isLoggedIn()
    };
  }
}

```

```

CRUDStore.addListener(EventsConsts.CHANGE, () => {
  this.setState({
    count: CRUDStore.getCount(),
  })
});

CRUDStore.addListener(EventsConsts.USER_EXISTS, () => {
  this.setState({errorUserExists: true});
});

CRUDStore.addListener(EventsConsts.DELETE_ERROR, () => {
  this.setState({errorDeletion: true});
});

CRUDStore.addListener(EventsConsts.UPDATE_ERROR, () => {
  this.setState({errorUpdating: true});
});

CRUDStore.addListener(EventsConsts.LOGIN_SUCCESS, () => {
  this.setState({isLogged: true});
});

CRUDStore.addListener(EventsConsts.LOGOUT, () => {
  this.setState({isLogged: false});
});

shouldComponentUpdate(newProps: Object, newState: State): boolean {
  return newState.addNew !== this.state.addNew ||
    newState.count !== this.state.count ||
    newState.errorUserExists !== this.state.errorUserExists ||
    newState.errorDeletion !== this.state.errorDeletion ||
    newState.errorUpdating !== this.state.errorUpdating ||
    newState.isLogged !== this.state.isLogged;
}

_addNewDialog() {
  this.setState({addNew: true});
}

_closeErrorDialog() {
  if (this.state.errorUserExists)
    this.setState({errorUserExists: false});

  if (this.state.errorDeletion)
    this.setState({errorDeletion: false});

  if (this.state.errorUpdating)
    this.setState({errorUpdating: false});
}

_addNew(action: string) {
  this.setState({addnew: false});
  if (action === 'confirm') {
    CRUDACTIONS.create(this.refs.form.getData());
  }
}

render() {

```

```

return (
  <div id="backOffice" style={{display: this.state.isLoggedIn ? 'block' :
  'none'}}>
    <div className="toolbar">
      <div className="toolbar-add">
        <Button
          onClick={this._addNewDialog.bind(this)}
          className="toolbar-add-btn">
          + add
        </Button>
      </div>
      <div className="toolbar-search">
        <input
          placeholder={this.state.count === 1
            ? 'Search 1 record...'
            : `Search ${this.state.count} records...`}
          onChange={CRUDActions.search.bind(CRUDActions)}
          onFocus={CRUDActions.startSearching.bind(CRUDActions)}/>
      </div>
    </div>
    <div className="back-office-datagrid">
      <Table />
    </div>
  {this.state.addNew
    ? <Dialog
      modal={true}
      header="Add New Player"
      confirmLabel="Add"
      onAction={this._addNew.bind(this)}>
      <Form ref="form"/>
    </Dialog>
    : null}
  {this.state.errorUserExists
    ? <Dialog
      modal={true}
      header="Error"
      confirmLabel="OK"
      hasCancel={false}
      onAction={this._closeErrorDialog.bind(this)}>
        User already exists!
    </Dialog>
    : null}
  {this.state.errorDeletion
    ? <Dialog
      modal={true}
      header="Error"
      confirmLabel="OK"
      hasCancel={false}
      onAction={this._closeErrorDialog.bind(this)}>
        Deleting player failed. Please try again later.
    </Dialog>
    : null}
  {this.state.errorUpdating
    ? <Dialog
      modal={true}>

```

```

        header="Error"
        confirmLabel="OK"
        hasCancel={false}
        onAction={this._closeErrorDialog.bind(this)}>
      Updating player data failed. Please try again later.
    </Dialog>
    : null}
  </div>
);
}

export default BackOffice

```

f) back-office/components/Button.js

```

import classNames from 'classnames';
import React from 'react';

const Button = props =>
  props.href
    ? <a {...props} className={classNames('button', props.className)}>
    : <button {...props} className={classNames('button',
  props.className)}>;
  
export default Button;

```

g) back-office/components/Dialog.js

```

import Button from './Button';
import React from 'react';

class Dialog extends React.Component {

  componentWillMount() {
    document.body.classList.remove('dialog-modal-open');
  }

  componentDidMount() {
    if (this.props.modal) {
      document.body.classList.add('dialog-modal-open');
    }
  }

  render() {
    return (
      <div className={this.props.modal ? 'dialog dialog-modal' : 'dialog'}>
        <div className={this.props.modal ? 'dialog-modal-wrap' : null}>
          <div className="dialog-header">{this.props.header}</div>
          <div className="dialog-body">{this.props.children}</div>
          <div className="dialog-footer">
            {this.props.hasCancel
              ? <span
                className="dialog-dismiss"
                onClick={this.props.onAction.bind(this, 'dismiss')}>
                  Cancel
                </span>
              : null}

```

```

    }
    <Button onClick={this.props.onAction.bind(this,
      this.props.hasCancel ? 'confirm' : 'dismiss')}>
      {this.props.confirmLabel}
    </Button>
  </div>
</div>
</div>
);
}
}

Dialog.defaultProps = {
  confirmLabel: 'OK',
  modal: false,
  onAction: () => {},
  hasCancel: true
};

export default Dialog;

```

h) back-office/components/EventsConsts.js

```

const EventsConsts = {
  'USER_EXISTS': 'userExists',
  'CHANGE': 'change',
  'DELETE_ERROR': 'deleteError',
  'UPDATE_ERROR': 'updateError',
  'LOGIN_SUCCESS': 'loginSuccess',
  'LOGIN_FAILED': 'loginFailed',
  'LOGOUT': 'logout'
};

export default EventsConsts;

```

i) back-office/components/Form.js

```

/* @flow */
import CRUDStore from '../flux/CRUDStore';
import FormInput from './FormInput';
import React, {Component} from 'react';
import type {FormInputField, FormInputFieldValue} from './FormInput';

type Props = {
  readOnly?: boolean,
  recordId: ?number,
};

class Form extends Component {
  fields: Array<Object>;
  initialData: ?Object;

  constructor(props: Props) {
    super(props);
    if (!this.props.fields) {
      this.fields = CRUDStore.getSchema();
    } else {
  
```

```

    this.fields = this.props.fields;
}

if ('recordId' in this.props) {
  this.initialData = CRUDStore.getRecord(this.props.recordId);
}
}

getData(): Object {
  let data: Object = {};
  this.fields.forEach((field: FormInputField) =>
    data[field.id] = this.refs[field.id].getValue()
  );
  return data;
}

render() {
  const { readOnly } = this.props;

  return (
    <form id="form">{this.fields.map((field: FormInputField) => {
      let prefilled: FormInputFieldValue = '';
      if (this.initialData) {
        prefilled = this.initialData[field.id];
        console.log('>>> prefilled:', prefilled);
      }

      if (!readOnly) {
        return (
          <div className="form-row" key={field.id}>
            <label className="form-label"
              htmlFor={field.id}>{field.label}</label>
            <FormInput {...field} ref={field.id} defaultValue={prefilled}
              readOnlyEmail={this.props.readOnlyEmail}/>
          </div>
        );
      }

      if (!prefilled) {
        return null;
      }

      return (
        <div className="form-row" key={field.id}>
          <span className="form-label">{field.label}</span>
          {
            <div>{prefilled}</div>
          }
        </div>
      );
    }), this}></form>
  );
}

export default Form;

```

j) back-office/components/FormInput.js

```
/* @flow */

import React from 'react';

type FormInputFieldType = 'number' | 'email' | 'text' | 'input';

export type FormInputFieldValue = string | number;

export type FormInputField = {
  type: FormInputFieldType,
  defaultValue?: FormInputFieldValue,
  id?: string,
  options?: Array<string>,
  label?: string,
};

class FormInput extends React.Component {

  props: FormInputField;

  getValue(): FormInputFieldValue {
    return 'value' in this.refs.input
      ? this.refs.input.value
      : this.refs.input.getValue();
  }

  render() {
    const common: Object = {
      id: this.props.id,
      ref: 'input',
      defaultValue: this.props.defaultValue,
    };

    switch (this.props.type) {
      case 'number':
        return (
          <input
            {...common}
            type="number"
            defaultValue={parseInt(this.props.defaultValue, 10) || 0} />
        );
      case 'password':
        return (
          <input
            {...common}
            type="password" />
        );
      case 'email':
        if (!this.props.readOnlyEmail) {
          return (
            <input
              {...common}
              type="email"
              defaultValue={this.props.defaultValue} />
          );
        }
        return (
          <input
```

```

{...common}
type="email"
readOnly="true"
defaultValue={this.props.defaultValue} />
);
case 'text':
return <textarea {...common} />;
default:
return <input {...common} type="text" />;
}
}

export default FormInput;

```

k) back-office/components/Login.js

```

import React from 'react';
import Button from './Button';
import EventsConsts from '../components/EventsConsts';
import SchemaEmail from '../schema-login-email-form';
import Form from './Form';
import CRUDStore from '../flux/CRUDStore';
import ViewManipulator from '../../utils/view-manipulator';

type State = {
  isLoggedIn: boolean,
};

class Login extends React.Component {
  state: State;

  constructor() {
    super();
    this.state = {
      isLoggedIn: CRUDStore.isLoggedIn()
    };
  }

  Login.login = Login.login.bind(this);

  CRUDStore.addListener(EventsConsts.LOGIN_SUCCESS, () => {
    this.setState({isLoggedIn: true});

    ViewManipulator.toggleErrorMessageView(document.querySelector('#errorMsg'),
      'Wrong login details.', false);
  });

  CRUDStore.addListener(EventsConsts.LOGIN_FAILED, () => {
    this.setState({isLoggedIn: false});

    ViewManipulator.toggleErrorMessageView(document.querySelector('#errorMsg'),
      'Wrong login details.', true);
  });

  CRUDStore.addListener(EventsConsts.LOGOUT, () => {
    this.setState({isLoggedIn: false});
  });
}

export default Login;

```

```

static login() {
  CRUDStore.checkLogin(this.refs.loginForm.getData());
}

render() {
  return <div style={{display: this.state.isLoggedIn ? 'none' : 'block'}}>
    <h3>Bingo Bigul Back Office Login</h3>
    <Form fields={SchemaEmail} ref="loginForm" />
    <Button onClick={Login.login}>
      Login
    </Button>
    <footer>
      <span>&copy; 2016{new Date().getFullYear() && "-" + new Date().getFullYear()} <a href="mailto: mihail.gaberov@gmail.com">Mihail Gaberov</a>.</span>
    </footer>
  </div>
}
}

export default Login;

```

I) back-office/components/Logo.js

```

import React from 'react';

class Logo extends React.Component {
  render() {
    return <div id="logo" />;
  }
}

export default Logo;

```

m) back-office/components/Logout.js

```

import React from 'react';
import Button from './Button';
import EventsConsts from '../components/EventsConsts';
import CRUDStore from '../flux/CRUDStore';

type State = {
  isLoggedIn: boolean
}

class Logout extends React.Component {

  state: State;

  constructor() {
    super();
    Logout.logout = Logout.logout.bind(this);
  }

  this.state = {
    isLoggedIn: CRUDStore.isLoggedIn()
  };

  CRUDStore.addListener(EventsConsts.LOGIN_SUCCESS, () => {
    this.setState({isLoggedIn: true});
  });
}

```

```

CRUDStore.addListener(EventsConsts.LOGOUT, () => {
  this.setState({isLoggedIn: false});
}

static logout() {
  CRUDStore.logout();
}

render() {
  return <div className="logout-module" style={{display: this.state.isLoggedIn
? 'block' : 'none'}}>
  <Button onClick={Logout.logout}>
    Logout
  </Button>
</div>
}
}

export default Logout;

```

n) back-office/components/Table.js

```

import * as Immutable from 'immutable';
import Actions from './Actions';
import CRUDActions from '../flux/CRUDActions';
import CRUDStore from '../flux/CRUDStore';
import Dialog from './Dialog';
import Form from './Form';
import FormInput from './FormInput';
import React, { Component } from 'react';
import classNames from 'classnames';
import invariant from 'invariant';
import EventsConsts from './EventsConsts';

type EditState = {
  row: number,
  key: string
};

type DialogState = {
  idx: number,
  type: string
};

type State = {
  data: Immutable.List<Object>,
  sortBy: ?string,
  descending: boolean,
  edit: ?EditState,
  dialog: ?DialogState
};

class Table extends Component {
  state: State;
  schema: Array<Object>;

  constructor() {
    super();
  }
}

```

```

this.state = {
  data: CRUDStore.getData(),
  sortBy: null, // schema.id
  descending: false,
  edit: null, // {row index, schema.id},
  dialog: null, // {type, idx}
};
this.schema = CRUDStore.getSchema();
CRUDStore.addListener(EventsConsts.CHANGE, () => {
  this.setState({
    data: CRUDStore.getData(),
  })
});
}

_sort(key: string) {
  const descending = this.state.sortBy === key && !this.state.descending;
  CRUDActions.sort(key, descending);
  this.setState({
    sortBy: key,
    descending: descending,
  });
}

_showEditor(e: Event) {
  const target = ((e.target: any): HTMLElement);
  this.setState({
    edit: {
      row: parseInt(target.dataset.row, 10),
      key: target.dataset.key,
    }
  });
}

_save(e: Event) {
  e.preventDefault();
  invariant(this.state.edit, 'Messed up edit state');
  CRUDActions.updateField(
    this.state.edit.row,
    this.state.edit.key,
    this.refs.input.getValue()
  );
  this.setState({
    edit: null,
  });
}

_actionClick(rowidx: number, action: string) {
  this.setState({ dialog: { type: action, idx: rowidx } });
}

_deleteConfirmationClick(action: string) {
  this.setState({ dialog: null });
  if (action === 'dismiss') {
    return;
  }
  const index = this.state.dialog && this.state.dialog.idx;
  invariant(typeof index === 'number', 'Unexpected dialog state');
  CRUDActions.delete(index);
}

```

```

_saveDataDialog(action: string) {
  this.setState({ dialog: null });
  if (action === 'dismiss') {
    return;
  }
  const index = this.state.dialog && this.state.dialog.idx;
  invariant(typeof index === 'number', 'Unexpected dialog state');
  CRUDActions.updateRecord(index, this.refs.form.getData());
}

render() {
  return (
    <div id="main-table">
      {this._renderTable()}
      {this._renderDialog()}
    </div>
  );
}

_renderDialog() {
  if (!this.state.dialog) {
    return null;
  }
  const type = this.state.dialog.type;
  switch (type) {
    case 'delete':
      return this._renderDeleteDialog();
    case 'info':
      return this._renderFormDialog(true);
    case 'edit':
      return this._renderFormDialog();
    default:
      throw Error(`Unexpected dialog type ${type}`);
  }
}

_renderDeleteDialog() {
  const index = this.state.dialog && this.state.dialog.idx;
  invariant(typeof index === 'number', 'Unexpected dialog state');
  const first = this.state.data.get(index);
  const nameGuess = first.name;
  return (
    <Dialog
      modal={true}
      header="Confirm"
      confirmLabel="Delete"
      onAction={this._deleteConfirmationClick.bind(this)}
    >
      {'Are you sure you want to delete "${nameGuess}"?'}
    </Dialog>
  );
}

_renderFormDialog(readonly: ?boolean) {
  const index = this.state.dialog && this.state.dialog.idx;
  invariant(typeof index === 'number', 'Unexpected dialog state');
  return (
    <Dialog
      modal={true}

```

```

        header={readonly ? 'Player Info' : 'Edit Player Data'}
        confirmLabel={readonly ? 'OK' : 'Save'}
        hasCancel={!readonly}
        onAction={this._saveDataDialog.bind(this)}
    >
    <Form
        ref="form"
        recordId={index}
        readonly={!readonly}
        readOnlyEmail={true} />
    </Dialog>
);
}

_renderTable() {
    return (
        <table>
            <thead>
                <tr>{
                    this.schema.map(item => {
                        if (!item.show) {
                            return null;
                        }
                        let title = item.label;
                        if (this.state.sortBy === item.id) {
                            title += this.state.descending ? '\u2191' : '\u2193';
                        }
                    return (
                        <th
                            className={`schema-${item.id}`}
                            key={item.id}
                            onClick={this._sort.bind(this, item.id)}
                        >
                            {title}
                        </th>
                    );
                }, this)
            >
                <th className="ExcelNotSortable">Actions</th>
            </tr>
        </thead>
        <tbody onDoubleClick={this._showEditor.bind(this)}>
            {this.state.data.map((row, rowidx) => {
                return (
                    <tr key={rowidx}>{
                        Object.keys(row).map((cell, idx) => {
                            const schema = this.schema[idx];
                            if (!schema || !schema.show) {
                                return null;
                            }

                            const edit = this.state.edit;
                            let content = row[schema.id];

                            if (edit && edit.row === rowidx && edit.key === schema.id && schema.editable) {
                                content = (
                                    <form onSubmit={this._save.bind(this)}>
                                        <FormInput ref="input" {...schema}>
                                            defaultValue={content} />
                                    </form>
                                );
                            }
                        })
                    </tr>
                );
            })
        </tbody>
    );
}

```

```

        </form>
    );
}
return (
<td
  className={classNames({
    ['schema-$\{schema.id\}']: true,
    'ExcelEditable': false,
    'ExcelDataLeft': schema.align === 'left',
    'ExcelDataRight': schema.align === 'right',
    'ExcelDataCenter': schema.align !== 'left' &&
schema.align !== 'right',
  })}
  key={idx}
  data-row={rowidx}
  data-key={schema.id}
  data-testid="td">
  {content}
</td>
);
}, this);
<td className="ExcelDataCenter" data-testid="td">
<Actions onAction={this._actionClick.bind(this, rowidx)} />
</td>
</tr>
);
}, this);
</tbody>
</table>
);
}
}

export default Table;

```

o) back-office/flux/CRUDActions.js

```

/* @flow */

import CRUDStore from './CRUDStore';
import { List } from 'immutable';

const CRUDActions = {
  create(newRecord: Object) {
    CRUDStore.addRecord(newRecord);
  },
  delete(recordId: number) {
    let data: List<Object> = CRUDStore.getData();
    CRUDStore.deleteRecord(data.get(recordId), recordId);
  },
  updateRecord(recordId: number, newRecord: Object) {
    CRUDStore.updateRecord(newRecord, recordId);
  },
  updateField(recordId: number, key: string, value: string | number) {
    let record = CRUDStore.getData().get(recordId);
    record[key] = value;
  }
};

export default CRUDActions;

```

```

    CRUDStore.updateRecord(record, recordId);
},
_preSearchData: null,
startSearching() {
  this._preSearchData = CRUDStore.getData();
},
search(e: Event) {
  const target = ((e.target: any): HTMLInputElement);
  const needle: string = target.value.toLowerCase();
  if (!needle) {
    CRUDStore.setData(this._preSearchData);
    return;
  }
  const fields = CRUDStore.getSchema().map(item => item.id);
  if (!this._preSearchData) {
    return;
  }
  const searchdata = this._preSearchData.filter(row => {
    for (let f = 0; f < fields.length; f++) {
      if (row[fields[f]] !== undefined) {
        if (row[fields[f]].toString().toLowerCase().indexOf(needle) > -1) {
          return true;
        }
      }
    }
    return false;
  });
  CRUDStore.setData(searchdata, /* commit */ false);
},
_sortCallback(a: (string | number), b: (string | number), descending: boolean): number {
  let res: number = 0;
  if (typeof a === 'number' && typeof b === 'number') {
    res = a - b;
  } else {
    res = String(a).localeCompare(String(b));
  }
  return descending ? -1 * res : res;
},
sort(key: string, descending: boolean) {
  CRUDStore.setData(CRUDStore.getData().sort(
    (a, b) => this._sortCallback(a[key], b[key], descending)
  ));
},
};

export default CRUDActions

```

p) back-office/flux/CRUDStore.js

```

/* @flow */
import { EventEmitter } from 'fbemitter';

```

```

import { List } from 'immutable';
import schema from '../schema';
import ApiCtrl from '../../API/api-controller';
import EventsConsts from '../components/EventsConsts';

let data: List<Object>;
const emitter = new EventEmitter();

const CRUDStore = {
  init(storage) {
    if (!storage) {
      let initialRecord = {};
      schema.forEach(item => initialRecord[item.id] = item.sample);
      data = List([initialRecord]);
    } else {
      data = List(storage);
    }
  },
  getData(): List<Object> {
    return data;
  },
  getSchema(): Array<Object> {
    return schema;
  },
  setData(newData: List<Object>, commit: boolean = true) {
    data = newData;
    emitter.emit(EventsConsts.CHANGE);
  },
  addListener(eventType: string, fn: Function) {
    emitter.addListener(eventType, fn);
  },
  getCount(): number {
    return data.count();
  },
  getRecord(recordId: number): ?Object {
    return data.get(recordId);
  },
  addRecord(objRecord: Object) {
    ApiCtrl.createPlayerPromise(objRecord).then((e) => {
      if (e.isExisted) {
        emitter.emit(EventsConsts.USER_EXISTS);
        return false;
      }

      if (!e.token) {
        return false;
      }

      CRUDStore.setData(CRUDStore.getData().unshift(objRecord));
      emitter.emit(EventsConsts.CHANGE);
    });
  },
}

```

```

deleteRecord(objRecord, recordId) {
  if (objRecord.email) {
    ApiCtrl.deletePlayerPromise(objRecord.email).then((e) => {
      if (e.ok) {
        CRUDStore.setData(CRUDStore.getData().remove(recordId));
        emitter.emit(EventsConsts.CHANGE);
      }
    });
  } else {
    emitter.emit(EventsConsts.DELETE_ERROR);
  }
}

updateRecord(objRecord, recordId) {
  let objUpdated = {};
  if (objRecord._id) {
    objUpdated.email = objRecord.email;
    objUpdated.name = objRecord.name;
    objUpdated.balance = objRecord.balance;
    objUpdated.wins = objRecord.wins;
  } else {
    objUpdated = objRecord;
  }
  if (objUpdated.email) {
    ApiCtrl.updatePlayerDataPromise(objUpdated).then((newRecord) => {
      if (newRecord) {
        CRUDStore.setData(CRUDStore.getData().set(recordId, newRecord));
        emitter.emit(EventsConsts.CHANGE);
      }
    });
  } else {
    emitter.emit(EventsConsts.UPDATE_ERROR);
  }
}

checkLogin(objCredentials) {
  if (!objCredentials.email || !objCredentials.password) {
    emitter.emit(EventsConsts.LOGIN_FAILED);
    return false;
  }

  ApiCtrl.loginAdminPromise(objCredentials).then((data) => {
    if (data.token) {
      emitter.emit(EventsConsts.LOGIN_SUCCESS);

      // Save the credentials to local storage
      localStorage.setItem('admin-token', data.token);
    } else {
      emitter.emit(EventsConsts.LOGIN_FAILED);
    }
  });
}

isLoggedIn() {
  let payload = null;

  let token = localStorage.getItem('admin-token');

  if (token !== null) {

```

```

payload = token.split('.')[1];
payload = window.atob(payload);
payload = JSON.parse(payload);

return payload.exp > Date.now() / 1000;
} else {
  return false;
}
},
logout() {
  localStorage.removeItem('admin-token');
  emitter.emit(EventsConsts.LOGOUT);
}
};

export default CRUDStore

```

q) back-office/back-office-app.js

```

import CRUDStore from './flux/CRUDStore';
import React from 'react';
import ReactDOM from 'react-dom';
import Logo from './components/Logo';
import Login from './components/Login';
import Logout from './components/Logout';
import BackOffice from './components/BackOffice';
import ApiCtrl from '../API/api-controller';

ApiCtrl.getPlayersDataPromise().then((data) => {
  CRUDStore.init(data);

  // Start the app when the data is fetched from the database
  startApp();
});

const startApp = () => {
  ReactDOM.render(
    <div>
      <div className="main">
        <div className="app-header">
          <Logo /> Bingo Bigul Back Office
          <Logout />
        </div>
        <BackOffice />
      </div>
      <div className="login-form">
        <Login />
      </div>
    </div>,
    document.getElementById('backOfficeApp')
  );
};

```

r) back-office/discoverer.js

```

import React from 'react';
import ReactDOM from 'react-dom';

```

```

import CRUDStore from './flux/CRUDStore';
import Button from './components/Button';
import Logo from './components/Logo';
import FormInput from './components/FormInput';
import Form from './components/Form';
import Actions from './components/Actions';
import Dialog from './components/Dialog';
import data from './dummy-data';

CRUDStore.init(data);

ReactDOM.render(
<div style={{ padding: '20px' }}>
  <h1>Discoverer Tool</h1>

  <h2>Logo</h2>
  <div style={{ display: 'inline-block', background: 'purple' }}>
    <Logo/>
  </div>

  <h2>Button</h2>
  <div>Button with onClick: <Button onClick={() => alert('Ouch!')}>Click me</Button></div>
  <div>A link: <Button href="http://mihail-gaberov.eu">Follow me</Button></div>
  <div>Custom class name: <Button className="custom">I do nothing</Button></div>

  <h2>Form inputs</h2>
  <table><tbody>
    <tr>
      <td>Vanilla input</td>
      <td><FormInput /></td>
    </tr>
    <tr>
      <td>Prefilled</td>
      <td><FormInput defaultValue="it's like a default" /></td>
    </tr>
    <tr>
      <td>Year</td>
      <td><FormInput type="year" /></td>
    </tr>
    <tr>
      <td>Vanilla textarea</td>
      <td><FormInput type="text" /></td>
    </tr>
  </tbody></table>

  <h2>Form</h2>
  <Form />

  <h2>Form readonly</h2>
  <Form readOnly={true} recordId={0} />

  <h2>Actions</h2>
  <div><Actions onAction={type => alert(type)}></div>

  <h2>Dialog</h2>
  <div><Dialog

```

```
header="Out of the box example"
onAction={type => alert(type)}>
Hello, dialog!
</Dialog></div>
<p>&nbsp;</p>
<div><Dialog
  header="No cancel, custom button"
  hasCancel={false}
  confirmLabel="Whatever"
  onAction={type => alert(type)}>
  Anything goes here, see: <Button>A button</Button>
</Dialog></div>
</div>,
document.getElementById('discoverer')
);
```

s) back-office/dummy-data.js

```
export default [
{
  name: 'Mihail Gaberov',
  email: 'mihail.gaberov@gmail.com',
  balance: '555',
  wins: '1232'
}]
```

t) back-office/schema.js

```
export default [
{
  id: 'email',
  label: 'Email',
  type: 'email',
  show: true,
  sample: 'me@mihail-gaberov.eu',
  editable: false
},
{
  id: 'name',
  label: 'Name',
  show: true,
  sample: 'Mihail Gaberov',
  align: 'left',
  editable: true
},
{
  id: 'password',
  label: 'Password',
  show: false,
  editable: true
},
{
  id: 'balance',
  label: 'Balance',
  type: 'number',
  show: true,
  sample: '50',
  step: 1
}]
```

```
    editable: true
  },
{
  id: 'wins',
  label: 'Wins',
  type: 'number',
  show: true,
  sample: '7',
  editable: true
}
]
```

u) **back-office/schema-login-email-form.js**

```
export default [
{
  id: 'email',
  label: 'Email',
  type: 'email',
  show: true,
  sample: 'me@mihail-gaberov.eu',
  editable: false
},
{
  id: 'name',
  label: 'Name',
  show: true,
  sample: 'Mihail Gaberov',
  align: 'left',
  editable: true
},
{
  id: 'password',
  label: 'Password',
  show: false,
  editable: true
},
{
  id: 'balance',
  label: 'Balance',
  type: 'number',
  show: true,
  sample: '50',
  editable: true
},
{
  id: 'wins',
  label: 'Wins',
  type: 'number',
  show: true,
  sample: '7',
  editable: true
}
]
```

v) blower/blower.js

```
import ViewManipulator from '../utils/view-manipulator';
import { paper } from '../../node_modules/paper/dist/paper-full';
import { NumbersGenerator } from '../utils/numbers-generator';
import { EventsConsts } from '../events/events-consts';

class Ball {
  constructor(point, vector) {
    if (!vector || vector.isZero()) {
      this.vector = {x: Math.random() * 5, y: Math.random() * 5};
    } else {
      this.vector = {x: vector.x * 2, y: vector.y * 2};
    }

    this.point = point;
    this.dampen = 0.4;
    this.gravity = 3;
    Ball.bounce = -0.6;

    let color = {
      hue: Math.random() * 360,
      saturation: 1,
      brightness: 1
    };
    let gradient = new paper.Gradient([color, 'black'], true);
    let radius = this.radius = 15;
    let ball = new paper.Group({
      children: [
        new paper.Path.Circle({
          radius: radius
        }),
        fillColor: new paper.Color(gradient, 0, radius, radius / 8)
      ];
    });

    let txt = new paper.PointText(this.center);
    txt.style = {
      justification: 'center',
      fontWeight: 'bold',
      fillColor: 'white'
    };

    txt.content = NumbersGenerator.getRandomNumber(1, 75);
    ball.addChild(txt);

    this.item = new paper.Group({
      children: [ball],
      transformContent: false,
      position: this.point
    });
  }

  iterate() {
    let size = paper.view.size;
    this.vector.y += this.gravity;
    this.vector.x *= 0.99;
    let pre = {
      x: this.point.x + this.vector.x,
      y: this.point.y + this.vector.y
    }
  }
}
```

```

};

if (pre.x < this.radius || pre.x > size.width - this.radius)
  this.vector.x *= -this.dampen / 2;

if (pre.y < this.radius || pre.y > size.height - this.radius) {
  if (Math.abs(this.vector.x) < 3) {
    this.vector = {
      x: Math.random() * 150,
      y: Math.random() * 320
    };
  }
  this.vector.y *= Ball.bounce / 2;
}

let max = paper.Point.max(this.radius, {
  x: this.point.x + this.vector.x,
  y: this.point.y + this.vector.y
});

this.item.position = this.point = paper.Point.min(max, {
  x: size.width - this.radius,
  y: size.height - this.radius
});
this.item.rotate(this.vector.x);
}

putBallsOnBottom() {
  let size = paper.view.size;
  this.vector.y += this.gravity;
  this.vector.x *= 0.99;
  let pre = {
    x: this.point.x + this.vector.x,
    y: this.point.y + this.vector.y
  };

  if (pre.x < this.radius || pre.x > size.width - this.radius)
    this.vector.x *= -this.dampen;

  if (pre.y < this.radius || pre.y > size.height - this.radius) {
    if (Math.abs(this.vector.x) < .3) {
      this.vector = {
        x: Math.random() * 150,
        y: Math.random() * 320
      };
    }
  }

  let max = paper.Point.max(this.radius, {
    x: this.point.x + this.vector.x,
    y: this.point.y + this.vector.y
  });

  this.item.position = this.point = paper.Point.min(max, {
    x: size.width - this.radius,
    y: size.height - this.radius
  });
  this.item.rotate(this.vector.x);
}

```

```

}

class Blower {
constructor(element) {
  document.addEventListener(EventsConsts.START_GAME, () => {
    this.startAnimation();
  });

  document.addEventListener(EventsConsts.END_GAME, () => {
    this.stopAnimation();
  });
  this.element = element;
  this.balls = [];
  this.init = {
    play: false,
    isPlaying: false
  };

  if (element) {
    element.setAttribute('width', '208px');
    element.setAttribute('height', '208px');
    paper.setup(element);
  } else {
    throw new Error('There is no canvas element to draw the blower in.');
  }

  for (let i = 0; i < 75; i++) {
    let position = {
      x: Math.random() * (paper.view.size.width - 1) + 1,
      y: Math.random() * (paper.view.size.height - 140) + 140
    },
    vector = new paper.Point((Math.random() - 0.5) * 50, Math.random() * 100),
    ball = new Ball(position, vector);

    this.balls.push(ball);
  }

  paper.view.onFrame = () => {
    for (let i = 0, l = this.balls.length; i < l; i++) {
      if (this.init.play) {
        this.balls[i].iterate();
      } else if (this.init.play === false && this.init.isPlaying === true) {
        this.balls[i].putBallsOnBottom();
      }
    }
  };
}

startAnimation() {
  ViewManipulator.toggleVisibility(this.element, true);
  for (let i = 0, l = this.balls.length; i < l; i++) {
    this.balls[i].point = {
      x: Math.random() * (paper.view.size.width - 10) + 10,
      y: Math.random() * (paper.view.size.height - 10) + 10
    };
  }
  this.init.play = true;
  this.init.isPlaying = true;
}

```

```

stopAnimation() {
  this.init.play = false;

  setTimeout(() => {
    this.init.isPlaying = false;
    ViewManipulator.toggleVisibility(this.element, false);
  }, 5000);
}

export default Blower;

```

w) card/card.js

```

import { EventsConsts } from '../events/events-consts';
import { WinningPatterns } from '../utils/winning-patterns';

class Card {
  constructor(objCard) {
    this.arrDrawnNumbers = [];
    this.divCard = document.createElement('div');
    this.divCard.setAttribute('id', 'card');
    this.divCard.setAttribute('class', 'col-xs-6 col-sm-3');
    this.divCard.innerHTML = '<table>' +
      '<tr>' +
      '<th class="firstCol">B</th>' +
      '<th class="secondCol">I</th>' +
      '<th class="thirdCol">N</th>' +
      '<th class="fourthCol">G</th>' +
      '<th class="fifthCol">O</th>' +
      '</tr>' +
      '<tr>' +
      '<td id="11">' + objCard.col1[0] + '</td>' +
      '<td id="21">' + objCard.col2[0] + '</td>' +
      '<td id="31">' + objCard.col3[0] + '</td>' +
      '<td id="41">' + objCard.col4[0] + '</td>' +
      '<td id="51">' + objCard.col5[0] + '</td>' +
      '</tr>' +
      '<tr>' +
      '<td id="12">' + objCard.col1[1] + '</td>' +
      '<td id="22">' + objCard.col2[1] + '</td>' +
      '<td id="32">' + objCard.col3[1] + '</td>' +
      '<td id="42">' + objCard.col4[1] + '</td>' +
      '<td id="52">' + objCard.col5[1] + '</td>' +
      '</tr>' +
      '<tr>' +
      '<td id="13">' + objCard.col1[2] + '</td>' +
      '<td id="23">' + objCard.col2[2] + '</td>' +
      '<td id="33">' + objCard.col3[2] + '</td>' +
      '<td id="43">' + objCard.col4[2] + '</td>' +
      '<td id="53">' + objCard.col5[2] + '</td>' +
      '</tr>' +
      '<tr>' +
      '<td id="14">' + objCard.col1[3] + '</td>' +
      '<td id="24">' + objCard.col2[3] + '</td>' +
      '<td id="34">' + objCard.col3[3] + '</td>' +
      '<td id="44">' + objCard.col4[3] + '</td>' +
      '<td id="54">' + objCard.col5[3] + '</td>' +
      '</tr>' +

```

```

'<tr>' +
'<td id="15">' + objCard.col1[4] + '</td>' +
'<td id="25">' + objCard.col2[4] + '</td>' +
'<td id="35">' + objCard.col3[4] + '</td>' +
'<td id="45">' + objCard.col4[4] + '</td>' +
'<td id="55">' + objCard.col5[4] + '</td>' +
'</tr>' +
'</table>';

this.divCard.addEventListener('click', (e) => {
  this.clickCell(e.target);
});

document.addEventListener(EventsConsts.NEW_BALL_DRAWN, (e) => {
  this.arrDrawnNumbers.push(parseInt(e.detail.drawnNumber));
});

this.arrWinningNumbers = [];

return this.divCard;
}

clickCell(element) {
  Card.markNumber(element);
  const clickedElementValue = parseInt(element.innerText);
  if (clickedElementValue && !isNaN(clickedElementValue)) {
    if (this.arrDrawnNumbers.indexOf(clickedElementValue) !== -1 &&
    element.classList.contains('drawn')) {
      this.arrWinningNumbers.push(element.id);
      Card.markDrawnNumber(element);

      let isBingo = false;
      if (WinningPatterns.checkHorizontalPattern(this.arrWinningNumbers)) {
        isBingo = true;
      } else if (WinningPatterns.checkVerticalPattern(this.arrWinningNumbers)) {
        isBingo = true;
      } else if (WinningPatterns.checkDiagonalPattern(this.arrWinningNumbers)) {
        isBingo = true;
      } else if (WinningPatterns.checkCornersPattern(this.arrWinningNumbers)) {
        isBingo = true;
      }

      if (isBingo) {
        const elTable =
          element.parentElement.parentElement.parentElement.parentElement;
        elTable.classList.add('bingo');

        // Dispatch new event when a Bingo is won
        const event = new CustomEvent(EventsConsts.BINGO, {
          detail: {
            time: new Date()
          }, bubbles: true, cancelable: true
        });
        this.divCard.dispatchEvent(event);
      }
    }
  }
}

```

```

static markNumber(el) {
  if (!el.classList.contains('drawn') && el.id !== '33')
    el.classList.toggle('marked');
}

static markDrawnNumber(el) {
  if (!el.classList.contains('drawn'))
    el.classList.add('drawn');
}

export default Card;

```

x) card/card-drawer.js

```

import Card from './card';
import { EventsConsts } from '../events/events-consts';
import ViewManipulator from '../utils/view-manipulator';

class CardDrawer {
  constructor(objCards, element) {
    document.addEventListener(EventsConsts.START_GAME, () => {
      CardDrawer.draw(objCards, element);
      ViewManipulator.toggleVisibility(element, true);
    });

    document.addEventListener(EventsConsts.END_GAME, () => {
      setTimeout(() => {
        ViewManipulator.toggleVisibility(element, false);
      }, 5000);
    });
  }

  static draw(objCards, element) {
    let countCards = Object.keys(objCards).length;
    let arrCards = [];

    for (let i = 0; i < countCards; i++) {
      arrCards.push(CardDrawer.generateCardTable(objCards['card' + (i+1)]));
    }

    // Clean the cards container first
    element.innerHTML = '';
    arrCards.forEach((el) => {
      element.appendChild(el);
    });
  }

  static generateCardTable(objCard) {
    return new Card(objCard);
  }
}

export default CardDrawer;

```

y) card/card-generator.js

```
import CardNumbersGenerator from './card-numbers-generator'

class CardGenerator {
  constructor(conf) {
    this.cardsGenerator = new CardNumbersGenerator(conf);
  }

  /**
   * Generates given number of cards with random numbers. One card by default.
   * @param count
   * @returns {{}}
   */
  generateCards(count = 1) {
    let objCards = {};
    while(count > 0) {
      objCards['card' + count] = this.cardsGenerator.generate();
      count--;
    }

    return objCards;
  }
}

export default CardGenerator;
```

z) card/card-numbers-generator.js

```
import { NumbersGenerator } from '../utils/numbers-generator';

class CardNumbersGenerator {
  constructor(conf) {
    if (conf !== undefined) {
      this.conf = conf;
      this.arrAmericanNumbers = this.conf.gameConf.numbers;
    }
  }

  /**
   * Generates a random card with 5 columns and theirs numbers accordingly
   * @returns {{col1: (*|Array), col2: (*|Array), col3: (*|Array), col4: (*|Array), col5: (*|Array)}}
   */
  generate() {
    let arrCol1 =
      NumbersGenerator.getColumnNumbers(this.arrAmericanNumbers.slice(0, 15));
    let arrCol2 =
      NumbersGenerator.getColumnNumbers(this.arrAmericanNumbers.slice(15, 30));
    let arrCol3 =
      NumbersGenerator.getColumnNumbers(this.arrAmericanNumbers.slice(30, 45));
    let arrCol4 =
      NumbersGenerator.getColumnNumbers(this.arrAmericanNumbers.slice(45, 60));
    let arrCol5 =
      NumbersGenerator.getColumnNumbers(this.arrAmericanNumbers.slice(60, 75));

    // Add 'free' field
    arrCol3[2] = (this.conf.gameConf.freeSpotImgPath !== undefined ?
      this.conf.gameConf.freeSpotImgPath : '');
}

return {
  'col1': arrCol1,
  'col2': arrCol2,
  'col3': arrCol3,
  'col4': arrCol4,
  'col5': arrCol5
};
}

export default CardNumbersGenerator;

```

aa) dauber/ball.js

```

import { Utils } from '../utils/utils';
import { EventsConsts } from '../events/events-consts';
import Animator from '../utils/animator';

class Ball {
  constructor(num, pubsub, skin = { "name": "original" }) {
    this.elBall = document.createElement('div');
    this.elBall.setAttribute('id', 'ball');
    this.elNumber = document.createElement('span');
    this.elNumber.innerText = num;
    this.elInnerCircle = document.createElement('div');
    this.elInnerCircle.setAttribute('id', 'innerCircle');
    this.elBall.className = `${skin.name}_${Utils.getCssClassByNumber(num)}`;
    this.pubsub = pubsub;
  }

  draw(parentElement, visibleBallNum, isSecondPhase = false) {
    this.elInnerCircle.appendChild(this.elNumber);
    this.elBall.appendChild(this.elInnerCircle);
    parentElement.appendChild(this.elBall);

    setTimeout(() => {
      if (isSecondPhase) {
        this.move(this.elBall, Animator.bounce, Animator.quad, 1000, 5);
      } else {
        this.move(this.elBall, Animator.bounce, Animator.quad, 1000,
visibleBallNum);
      }
    }, 200);
  }

  move(element, delta1, delta2, duration = 1000, visibleBallNum) {
    const posUp = 2;
    const startPos = 62;
    const endPosBall2 = 47;
    const endPosBall3 = 32;
    const endPosBall4 = 17;
    const endPosBall5 = 2;

    // If there is the first animation - run it
    if (delta1 !== null) {
      this.animate({
        delay: 10,

```

```

duration: duration,
delta: delta1,
step: (delta) => {
  element.style.top = (-posUp * delta) + 7 + "%";
}
};

// If there is the second animation - run it
if (delta2 !== null) {
  setTimeout(() => {
    this.animate({
      delay: 10,
      duration: 1000,
      delta: delta2,
      isFifth: visibleBallNum,
      step: (delta) => {
        switch (visibleBallNum) {
          case 1:
            element.style.left = (-(startPos * delta) + startPos) + "%";
            Animator.rotateElement(element, -1440, Animator.linear, 400);
            break;
          case 2:
            element.style.left = (-(endPosBall2 * delta) + startPos) + "%";
            Animator.rotateElement(element, -1080, Animator.linear, 400);
            break;
          case 3:
            element.style.left = (-(endPosBall3 * delta) + startPos) + "%";
            Animator.rotateElement(element, -720, Animator.linear, 400);
            break;
          case 4:
            element.style.left = (-(endPosBall4 * delta) + startPos) + "%";
            Animator.rotateElement(element, -360, Animator.linear, 400);
            break;
          case 5:
            element.style.left = (-(endPosBall5 * delta) + startPos) + "%";
            Animator.rotateElement(element, -360, Animator.linear, 400);
            break;
        }
      }
    });
  }, 1000);
}

animate(opts) {
  const start = new Date();

  const id = setInterval(() => {
    let timePassed = new Date - start;
    let progress = timePassed / opts.duration;

    if (progress > 1)
      progress = 1;

    let delta = opts.delta(progress);
    opts.step(delta);

    if (progress === 1) {
      if (opts.isFifth === 5)

```

```

    this.pubsub.publish(EventsConsts.FIFTH_BALL_DRAWN, {});
    clearInterval(id);
}
}, opts.delay || 10);
}

export default Ball;

```

bb) dauber/dauber.js

```

import ViewManipulator from '../utils/view-manipulator';
import { EventsConsts } from '../events/events-consts';
import { NumbersGenerator } from '../utils/numbers-generator';
import PubSubService from '../events/publish-service';
import Ball from './ball';
import Animator from '../utils/animator';

class Dauber {
  constructor(conf = null, element) {
    if (conf !== null) {
      this.gameStarted = false;
      document.addEventListener(EventsConsts.START_GAME, () => {
        const drawBallTime = conf.gameConf.drawIntervalSeconds * 1000;
        this.startDrawing(drawBallTime);
        this.gameStarted = true;
      });
    }

    document.addEventListener(EventsConsts.END_GAME, () => {
      clearInterval(this.drawTimeout);
      this.drawTimeout = null;
      document.removeEventListener(EventsConsts.END_GAME, this.endGame);

      setTimeout(() => {
        ViewManipulator.toggleVisibility(this.element.parentElement, false);

        // Clear all game elements values
        this.arrDrawnNums = [];
        this.arrDrawnNums.length = 0;
        this.arrVisibleBalls = [];
        this.arrVisibleBalls.length = 0;
        this.visibleBallNum = 0;
        this.element.innerHTML = '';
        this.isSecondPhase = false;
      }, 5000);
    });
  }

  this.conf = conf;
  this.element = element;
  this.arrDrawnNums = [];
  this.drawTimeout = null;
  this.visibleBallNum = 0;
  this.arrVisibleBalls = [];
  this.isSecondPhase = false;
  this.pubsub = new PubSubService();
  this.pubsub.subscribe(EventsConsts.FIFTH_BALL_DRAWN, () => {
    this.animateVisibleBalls() );
    this.elVisibleBallsContainer = document.createElement('div');
  });
}

```

```

this.elVisibleBallsContainer.setAttribute('id', 'elVisibleBallsContainer');
this.element.appendChild(this.elVisibleBallsContainer);
} else {
  throw new Error('Dauber initialization error - no config');
}
}

doDraw() {
  const drawnNum = this.drawNewNumber();
  let ball = new Ball(drawnNum, this.pubsub, this.conf.gameConf.skin);
  ball.draw(this.element, ++this.visibleBallNum, this.isSecondPhase);
  this.arrVisibleBalls.push(ball);

  // Dispatch new event with the drawn number
  const event = new CustomEvent(EventsConsts.NEW_BALL_DRAWN, {
    detail: {
      drawnNumber: drawnNum,
      time: new Date()
    },
    bubbles: true, cancelable: true
  });
  this.element.dispatchEvent(event);

  if (this.visibleBallNum === 5) {
    this.visibleBallNum = 0;
    this.isSecondPhase = true;
  }

  // Set default value for turns count if no configured
  if (this.conf.gameConf.turnsCount === undefined)
    this.conf.gameConf.turnsCount = 23;

  if (this.arrDrawnNums.length >= this.conf.gameConf.turnsCount) {
    // End Game
    this.endGame();
  }
}

startDrawing(intervalinMs = 7000) {
  ViewManipulator.toggleVisibility(this.element.parentElement, true);
  this.drawTimeout = setInterval(() => {
    this.doDraw();
  }, intervalinMs);
}

endGame() {
  if (this.gameStarted) {
    // Dispatch new event when the game ends
    const event = new CustomEvent(EventsConsts.END_GAME, {
      detail: {
        time: new Date()
      },
      bubbles: true, cancelable: true
    });
    this.element.dispatchEvent(event);
    this.gameStarted = false;
  }
}

drawNewNumber() {

```

```

const randomIdx = NumbersGenerator.getRandomNumber(0,
this.conf.gameConf.numbers.length - 1);
const num = this.conf.gameConf.numbers[randomIdx];
if (num !== undefined) {
  if (this.arrDrawnNums.indexOf(num) !== -1) {
    // This number is already drawn - get another
    return this.drawNewNumber();
  } else {
    this.arrDrawnNums.push(num);
    return num;
  }
} else {
  throw new Error('Dauber draws undefined number');
}

animateVisibleBalls() {
  if (this.arrVisibleBalls.length > 0) {
    ViewManipulator.toggleVisibility(this.arrVisibleBalls[0].elBall, false);
    this.arrVisibleBalls.shift(); // remove the first drawn ball from the array
    this.elVisibleBallsContainer.style.left = '0';
    if (this.elVisibleBallsContainer.lastChild) {

      this.elVisibleBallsContainer.removeChild(this.elVisibleBallsContainer.lastChild)
    }

    this.arrVisibleBalls.forEach((ball) => {
      ball.elBall.style.left = (parseInt(ball.elBall.style.left) - 15) + '%';
      Animator.rotateElement(ball.elBall, 360, Animator.linear, 500);
    });

    for (let i = 0, len = this.arrVisibleBalls.length; i < len; i++) {
      this.elVisibleBallsContainer.appendChild(this.arrVisibleBalls[i].elBall);
    }

    Animator.moveVerticalHorizontal(this.elVisibleBallsContainer, 0, -15,
    Animator.quad, 500, '%');
  }
}

export default Dauber;

```

cc) events/events-consts.js

```

const EventsConsts = {
  LOGOUT: 'logout',
  NEW_BALL_DRAWN: 'newBallDrawn',
  FIFTH_BALL_DRAWN: 'fifthBallDrawn',
  START_GAME: 'startGame',
  END_GAME: 'endGame',
  BINGO: 'bingo',
  PRIZE_WON: 'prizeWon',
  FLYING_PRIZE_ANIMATION_ENDS: 'flyingPrizeAnimationEnds',
  ENOUGH_BALANCE: 'enoughBalance',
  NOT_ENOUGH_BALANCE: 'notEnoughBalance'
};

```

```
export { EventsConsts };
```

dd) events/pubsub-service.js

```
/*
 * Used pubsub pattern - https://davidwalsh.name/pubsub-javascript
 */
class PubSubService {
    constructor() {
        this.topics = {};
        this.hOP = this.topics.hasOwnProperty;
        this.idx = 0;
    }

    subscribe(topic, listener) {
        if (!this.hOP.call(this.topics, topic))
            this.topics[topic] = [];

        this.idx = this.topics[topic].push(listener) - 1;
    }

    remove(topic) {
        delete this.topics[topic][this.idx];
    }

    publish(topic, info) {
        if(!this.hOP.call(this.topics, topic))
            return;

        this.topics[topic].forEach((item) => {
            item(info !== undefined ? info : {});
        });
    }
}

export default PubSubService;
```

ee) initializer/initializer.js

```
import ApiController from '../API/api-controller';
import ViewManipulator from '../utils/view-manipulator';
import CardGenerator from '../card/card-generator';
import CardDrawer from '../card/card-drawer';
import MarketCards from '../market-place/market-cards';
import Blower from '../blower/blower';
import Dauber from '../dauber/dauber';
import Timer from '../utils/timer';
import {EventsConsts} from '../events/events-consts';
import WinningDialog from '../winning/winning-dialog';
import WinPatternsAnimModule from '../winning/win-patterns-anim-module';

class Initializer {
    static applyConfigurations(conf) {
        Initializer.setTitle(conf.gameConf.appTitle);
        Initializer.addWinningDialog(conf.gameConf.winningDialog);
        Initializer.setCardPrices(conf.gameConf.cardPrice);
    }
}
```

```

Initializer.addWinPatternAnimModule(conf.gameConf.winPatternsAnimModule);

Initializer.attachEnoughBalanceListener(Initializer.addMarketPlace(conf.gameConf.marketCards), conf, document);
    Initializer.addStartButton(conf,
Initializer.addMarketPlace(conf.gameConf.marketCards));
    Initializer.addDauber(conf);
    Initializer.addLogoutBtn();
    Initializer.showUserInfo();
}

static setTitle(appTitle) {
    document.querySelector('title').innerText = appTitle;
}

static addWinningDialog(isConfigured) {
    return isConfigured ? new WinningDialog('#winningDialogContainer') :
undefined;
}

static setCardPrices(price) {
    return MarketCards.setCardPrices(price,
document.querySelectorAll('.cards'));
}

static addWinPatternAnimModule(isConfigured) {
    let elWinPatternsAnimModule = undefined;
    if (isConfigured) {
        elWinPatternsAnimModule =
document.querySelector('#winPatternsAnimModule');
        new
WinPatternsAnimModule(elWinPatternsAnimModule.querySelector('#horizontal'),
5, 5, 'horizontal');
        new
WinPatternsAnimModule(elWinPatternsAnimModule.querySelector('#vertical'), 5,
5, 'vertical');
        new
WinPatternsAnimModule(elWinPatternsAnimModule.querySelector('#diagonal'), 5,
5, 'diagonal');
    }
    return elWinPatternsAnimModule;
}

static addStartButton(conf, elMarketPlace) {
let startBtn = undefined;
if (conf.gameConf.mainGame) {
    startBtn = document.querySelector('#startBtn');
    if (startBtn) {
        startBtn.addEventListener('click', () => {
            Initializer.buyCards(conf, elMarketPlace);
        });
    }
}
return startBtn;
}

static attachEnoughBalanceListener(elMarketPlace, conf, document) {
document.addEventListener(EventsConsts.ENOUGH_BALANCE, () => {
    ViewManipulator.toggleVisibility(elMarketPlace, false);
}

```

```

    ViewManipulator.toggleVisibility(document.querySelector('#footer'),
false);
    Initializer.getTimer(conf).startCounting();
}
}

static getTimer(conf) {
    return new Timer(
        document.querySelector('#timerContainer'),
        conf.gameConf.beforeStartGameSeconds,
        EventsConsts.START_GAME, true
    );
}

static addMarketPlace(isConfigured) {
    const elMarketPlace = document.querySelector('#marketPlace');
    if (!isConfigured) {
        ViewManipulator.toggleVisibility(elMarketPlace, false);
    }
    return elMarketPlace
}

static buyCards(conf, container) {
    if (conf.gameConf.playingCards) {
        const cardGen = new CardGenerator(conf);
        const marketCards = new MarketCards(container);
        MarketCards.buyCards(Initializer.renderPurchasedCards(marketCards,
cardGen), conf.gameConf.cardPrice);
    }
}

static renderPurchasedCards(marketCards, cardGen) {
    const purchasedCardsCount =
MarketCards.getPurchasedCardsCount(marketCards.getRadioButtonsArray());
    new CardDrawer(
        cardGen.generateCards(purchasedCardsCount),
        document.querySelector('#leftGameScreen')
    );
    return purchasedCardsCount;
}

static addDauber(conf) {
    let dauberElements = {};
    if (conf) {
        dauberElements.dauber = new Dauber(conf,
document.querySelector('#tube'));
        dauberElements.blower = new Blower(document.querySelector('#blower-
balloon'));
    }
    return dauberElements;
}

static showUserInfo() {
    if (ApiController.isLoggedIn()) {
        ViewManipulator.showUserInfo();
    }
}

static addLogoutBtn() {
    const apiCtrl = new ApiController();

```

```

const logoutBtn = document.querySelector('#logoutBtn');
if (logoutBtn) {
  logoutBtn.addEventListener('click', () => {
    apiCtrl.logout();
  });
}
}

export default Initializer;

```

ff) local-storage/local-storage-service.js

```

class LocalStorageService {

  static saveToken(token) {
    window.localStorage['mean-token'] = token;
  }

  static saveBalance(balance) {
    window.localStorage['balance'] = balance;
  }

  static getBalance() {
    if (window.localStorage['balance'] !== undefined)
      return window.localStorage['balance'];
  }

  static getToken() {
    if (window.localStorage['mean-token'] !== undefined)
      return window.localStorage['mean-token'];
  }

  static getAdminToken() {
    if (window.localStorage['admin-token'] !== undefined)
      return window.localStorage['admin-token'];
  }

  static logout() {
    window.localStorage.removeItem('mean-token');
  }

  static isLoggedIn() {
    let token = LocalStorageService.getToken();

    let payload;

    if (token) {
      payload = token.split('.')[1];
      payload = window.atob(payload);
      payload = JSON.parse(payload);

      return payload.exp > Date.now() / 1000;
    } else {
      return false;
    }
  }
}

```

```

static currentUser() {
  if (LocalStorageService.isLoggedIn()) {
    let token = LocalStorageService.getToken();
    let payload = token.split('.')[1];
    payload = window.atob(payload);
    payload = JSON.parse(payload);
    return {
      email : payload.email,
      name : payload.name,
      balance : payload.balance
    };
  }
}

export default LocalStorageService;

```

gg) market-place/market-cards.js

```

import { EventsConsts } from '../events/events-consts';
import ViewManipulator from '../utils/view-manipulator';
import ApiController from '../API/api-controller';

class MarketCards {
  constructor(container) {
    this.container = container;

    document.addEventListener(EventsConsts.END_GAME, () => {
      setTimeout(() => {
        ViewManipulator.toggleVisibility(container, true);
        ViewManipulator.toggleVisibility(document.querySelector('#footer'), true);
      }, 5000);
    });
  }

  getRadioButtonsArray() {
    return this.container.querySelectorAll('input[type=radio]');
  }

  static getPurchasedCardsCount(arrRadioButtons) {
    let numberOfCards = 0;
    let len = arrRadioButtons.length - 1;

    while (len >= 0) {
      if (arrRadioButtons[len].checked) {
        numberOfCards = Number(arrRadioButtons[len].value);
      }
      len--;
    }

    return numberOfCards;
  }

  static setCardPrices(price, arrElements) {
    let len = arrElements.length - 1;

    while (len >= 0) {
      const howManyCards =
        arrElements[len].querySelector('input[type=radio]').value;
        arrElements[len].querySelector('.price').innerHTML = '<i class="price-'
    }
  }
}

```

```

icon"></i>${parseInt(price * howManyCards)}';
    len--;
}

return arrElements;
}

static buyCards(count, price) {
    const totalSpent = Number(count) * Number(price);

    const promiseBalance = ApiController.getPlayerBalancePromise();
    promiseBalance.then((val) => {
        // Calculate if the user has enough money to buy the selected cards in order
        to start the game
        if (Number(val) >= 0) {
            ApiController.setNewBalance(totalSpent, true);
            const event = new CustomEvent(EventsConsts.ENOUGH_BALANCE, {
                detail: {
                    time: new Date()
                }, bubbles: true, cancelable: true
            });
            document.dispatchEvent(event);
        } else {
            // Dispatch new event for showing the not enough money dialog
            const event = new CustomEvent(EventsConsts.NOT_ENOUGH_BALANCE, {
                detail: {
                    time: new Date()
                }, bubbles: true, cancelable: true
            });
            document.dispatchEvent(event);
        }
    });
}

export default MarketCards;

```

hh) utils/animator.js

```

import { EventsConsts } from '../events/events-consts';

class Animator {

    static bounce(progress) {
        for (var a = 0, b = 1; 1; a += b, b /= 2) {
            if (progress >= (7 - 4 * a) / 11) {
                return -Math.pow((11 - 6 * a - 11 * progress) / 4, 2) + Math.pow(b, 2)
            }
        }
    }

    static quad(progress) {
        return Math.pow(progress, 2)
    }

    static linear(progress) {
        return progress
    }
}

```

```

}

static animate(opts) {
  const start = new Date;

  var id = setInterval(function () {
    const timePassed = new Date - start;
    let progress = timePassed / opts.duration;

    if (progress > 1) progress = 1;

    const delta = opts.delta(progress);

    opts.step(delta);

    if (progress === 1) {
      clearInterval(id);
    }
  }, opts.delay || 10);
}

static moveVerticalHorizontal(element, toTop, toLeft, delta, duration, units) {
  if (!isNaN(toTop)) {
    Animator.animate({
      delay: 5,
      duration: duration || 500,
      delta: delta,
      step: (delta) => {
        element.style.top = toTop * delta + units;
      }
    });
  }

  if (!isNaN(toLeft)) {
    Animator.animate({
      delay: 10,
      duration: duration || 1000,
      delta: delta,
      step: (delta) => {
        element.style.left = toLeft * delta + units;
      }
    });
  }
}

static rotateElement(element, degrees, delta, duration) {
  Animator.animate({
    delay: 10,
    duration: duration || 1000,
    delta: delta || Animator.linear,
    step: (delta) => {
      element.style.transform = "rotate(" + degrees * delta + "deg)";
    }
  });
}

static moveDiagonally(element, fromTop, fromLeft, toTop, toLeft, delta,
duration, units) {
  if (!isNaN(toLeft) && !isNaN(toTop)) {
    Animator.animate({

```

```

    delay: 10,
    duration: duration || 1000,
    delta: delta,
    step: (delta) => {
      element.style.top = fromTop + (toTop * delta) + units;
      element.style.left = fromLeft + (toLeft * delta) + units;
    }
  );
}

Animator.dispatchEventAfterDuration(EventsConsts.FLYING_PRIZE_ANIMATION_ENDS, duration);
}

static dispatchEventAfterDuration(eventName, duration = 1000) {
  // Dispatch new event when the animation duration expires
  setTimeout(() => {
    const event = new CustomEvent(eventName, {
      detail: {
        time: new Date()
      },
      bubbles: true,
      cancelable: true
    });
    document.dispatchEvent(event);
  }, duration);
}

return false;
}

export default Animator;

```

ii) utils/bangup.js

```

class Bangup {
  constructor(container, fromSum, toSum, duration) {
    this.container = container;
    this.fromSum = Math.round(fromSum);
    this.toSum = Math.round(toSum);
    this.startTime = null;
    this.duration = Number(duration) * 1000 || 2000;

    /**
     * http://paulirish.com/2011/requestanimationframe-for-smart-animating
     * shim layer with setTimeout fallback
     */
    window.requestAnimFrame = (function () {
      return window.requestAnimationFrame ||
        window.webkitRequestAnimationFrame ||
        window.mozRequestAnimationFrame ||
        window.oRequestAnimationFrame ||
        window.msRequestAnimationFrame ||
        function (callback) {
          window.setTimeout(callback, 1000 / 60);
        };
    })();
  }

  window.requestAnimationFrame(this.pileNumbers.bind(this));
}

```

```

    }

    pileNumbers(timestamp) {
      if (!this.startTime) {
        this.startTime = timestamp;
      }

      let progress = timestamp - this.startTime;
      this.container.innerHTML = Math.round(Bangup.easeOutExpo(progress,
      this.fromSum, this.toSum - this.fromSum, this.duration));

      // Continue until the end sum is reached
      if (Number(this.container.innerHTML) !== this.toSum) {
        window.requestAnimationFrame(this.pileNumbers.bind(this));
      }
    }

    /**
     * Robert Penner's easeOutExpo
     * @param currentTime current time
     * @param startValue start value
     * @param endValue end value – the value which should reach
     * @param duration duration in milliseconds
     * @returns {number}
     */
    static easeOutExpo(currentTime, startValue, endValue, duration) {
      return (currentTime == duration) ? startValue + endValue : endValue * (-
      Math.pow(2, -10 * currentTime / duration) + 1) + startValue;
    }
  }

  export default Bangup;

```

jj) utils/numbers-generator.js

```

const NumbersGenerator = {
  getRandomNumber (min, max) {
    if (max === undefined) {
      max = min;
      min = 0;
    }
    return Math.floor(Math.random() * (max - min + 1)) + min;
  },

  /**
   * Do random generation of the numbers by columns:
   * On a real Bingo card, each column has a different
   * range of numbers: B is 1–15, I is 16–30, N is 31–45,
   * G is 46–60, and O is 61–75
   * @returns {Array}
   */
  getColumnNumbers(arrNumbers, countNums = 5) {
    let objAdded = {}, arrOutput = [], i, l;

    for(i = 0, l = arrNumbers.length; i < l; ++i) {
      let randomNum = this.getRandomNumber(arrNumbers[1] , arrNumbers[14]);

      if(objAdded.hasOwnProperty(randomNum) || arrOutput.length === countNums)
    }
  }
}

```

```

        continue;
    }

    arrOutput.push(randomNum);
    objAdded[randomNum] = 1;
}

return arrOutput;
}

};

export { NumbersGenerator };

```

kk) utils/timer.js

```

class Timer {
constructor(element, seconds, eventName, isVisible) {
    this.element = element;
    this.seconds = seconds;
    this.eventName = eventName;
    this.isVisible = isVisible;
}

startCounting() {
let sec = 0;
if (!isNaN(parseInt(this.seconds))) {
    const intervalID = setInterval(() => {
        this.element.style.display = (this.isVisible ? 'block' : 'none');
        this.element.children[0].innerHTML = `00:0${this.seconds - sec}`;
        if (sec++ === this.seconds) {
            window.clearInterval(intervalID);
            this.triggerEvent();
            this.hide();
        }
    }, 1000);
}
}

hide() {
    this.element.style.display = 'none';
}

triggerEvent() {
const event = new CustomEvent(
    this.eventName,
    {
        detail: {
            time: new Date()
        },
        bubbles: true,
        cancelable: true
    }
);
this.element.dispatchEvent(event);
}

export default Timer;

```

II) utils/utils.js

```
const Utils = {
  eliminateDuplicates(arr) {
    return [...new Set(arr)];
  },
  /**
   * Give the color of a ball according to its number
   * which relates to its column. Bingo column colors are:
   * B – dark red, I – dark blue, N – dark purple, G – dark orange
   * O – dark green
   *
   * B is 1–15, I is 16–30, N is 31–45,
   * G is 46–60, and O is 61–75
   */
  getCssClassByNumber(num) {
    if (num === undefined) {
      throw new Error('Undefined number given to define ball color');
    }

    if (num >= 1 && num <= 15) {
      return 'ballB';
    } else if (num > 15 && num <= 30) {
      return 'ballI';
    } else if (num > 30 && num <= 45) {
      return 'ballN';
    } else if (num > 45 && num <= 60) {
      return 'ballG';
    } else if (num > 60 && num <= 75) {
      return 'ballO';
    }
  },
  countInArray(array, what) {
    return array.filter(item => item == what).length;
  }
};

export { Utils };
```

mm) utils/view-manipulator.js

```
import { EventsConsts } from '../events/events-consts';
import ApiController from '../API/api-controller';
import LocalStorageService from '../local-storage/local-storage-service';
import Bangup from '../utils/bangup';

class ViewManipulator {
  constructor(pubsub) {
    this.signInLink = document.querySelector('.sign-in-link a');
    this.registerForm = document.querySelector('#registerForm');
    this.loginForm = document.querySelector('#loginForm');
    this.marketPlace = document.querySelector('#marketPlace');
    this.btnCloseAlertMsg = document.querySelector('#btnCloseAlertMsg');
    this.isOnLoginPage = true;

    this.attachViewListeners();
    pubsub.subscribe(EventsConsts.LOGOUT, (eventData) => {
```

```

    if (eventData.isLogout) {
      ViewManipulator.updateViewState(undefined, undefined,
        LocalStorageService.isLoggedIn());
    }
  });

attachViewListeners() {
  if (this.signInLink)
    this.signInLink.addEventListener('click', () => {
      this.toggleFormView(this.isOnLoginPage);
    });

  // Registration form
  if (this.registerForm) {
    this.registerForm.addEventListener('submit', (e) => {
      e.preventDefault();
      ApiController.register();
    });
  }

  // Login form
  if (this.loginForm) {
    this.loginForm.addEventListener('submit', (e) => {
      e.preventDefault();
      ApiController.login();
    });
  }

  // We don't use jQuery => cannot use the Bootstrap implementation of closing
  // the
  // alert messages => implement it custom here
  if (this.btnCloseAlertMsg) {
    this.btnCloseAlertMsg.addEventListener('click', (e) => {
      ViewManipulator.toggleVisibility(e.target.parentNode.parentNode, false);
    });
  }

  // Market cards
  if (this.marketPlace) {
    this.marketPlace.addEventListener('click', (e) => {
      ViewManipulator.toggleMarketCardSelectedState(e);
    });
  }
}

toggleFormView(isOnLoginPage) {
  ViewManipulator.toggleVisibility(this.loginForm, !isOnLoginPage);
  ViewManipulator.toggleVisibility(this.registerForm, isOnLoginPage);
  if (!isOnLoginPage) {
    this.signInLink.innerText = 'Don\'t have an account? Register here!';
    this.isOnLoginPage = true;
  } else {
    this.signInLink.innerText = 'Already have an account? Sign in here!';
    this.isOnLoginPage = false;
  }
}

static toggleErrorMessageView(elAlertMsg, msg, isShow) {

```

```

elAlertMsg.querySelector('#messageText').innerText = msg;
ViewManipulator.toggleVisibility(elAlertMsg, isShow);
}

static updateViewState(elGameWrapper =
document.querySelector('#gameWrapper'), elRegisterPage =
document.querySelector('#registerPage'), isLoggedIn = false) {
  ViewManipulator.toggleVisibility(elGameWrapper, isLoggedIn);
  ViewManipulator.toggleVisibility(elRegisterPage, !isLoggedIn);
}

static showUserInfo(elUserProfile = document.querySelector('#userProfile')) {
  if (elUserProfile) {
    const elName = elUserProfile.querySelector('h2');
    const elEmail = elUserProfile.querySelector('div a');
    const objUserInfo = ApiController.getProfileInfo();
    elEmail.setAttribute('href', 'mailto:' + objUserInfo.email);
    elName.innerHTML = objUserInfo.name;
    elEmail.innerHTML = objUserInfo.email;

    const promiseBalance = ApiController.getPlayerBalancePromise();
    promiseBalance.then((val) => {
      ViewManipulator.updateBalance(0, val);
    });
  }
}

static updateBalance(fromSum, toSum) {
  const elBalance = document.querySelector('#userProfile').querySelector('h4 span');
  const balanceBangup = new Bangup(elBalance, fromSum, toSum);
}

static toggleMarketCardSelectedState(event) {
  const parent = event.target.parentNode.parentNode;
  const strSelectedClassName = 'selected';
  const strDesignatedClassName = 'jsMarketCardContainer';

  // Remove 'selected' class from all others elements
  const arrAllCards = Array.from(document.querySelectorAll('.' + strDesignatedClassName));
  let len = arrAllCards.length - 1;
  while (len >= 0) {
    if (typeof arrAllCards[len] !== 'undefined') {
      arrAllCards[len].classList.remove(strSelectedClassName);
    }
    len--;
  }

  if (parent.classList.contains(strDesignatedClassName)) {
    parent.classList.add(strSelectedClassName);
  } else {
    parent.parentNode.classList.add(strSelectedClassName);
  }
}

static toggleVisibility(element, isVisible) {
  element.style.display = (isVisible ? 'block' : 'none');
}

```

```

}

export default ViewManipulator;

```

nn) **utils/winning-patterns.js**

```

import { Utils } from '../utils/utils';

const WinningPatterns = {
  isBingo(arrCoordinates) {
    let isBingo = false;
    for (let i = 1; i <= 5; i++) {
      if (i !== 3) {
        if (Utils.countInArray(arrCoordinates, i) === 5) {
          isBingo = true;
        }
      } else {
        if (Utils.countInArray(arrCoordinates, i) === 4) {
          isBingo = true;
        }
      }
    }
    return isBingo;
  },
  checkHorizontalPattern(arr) {
    const arrCoordinates = [];
    arr.forEach((el) => {
      let coorY = el.substr(1, 2);
      arrCoordinates.push(coorY);
    });
    return this.isBingo(arrCoordinates);
  },
  checkVerticalPattern(arr) {
    const arrCoordinates = [];
    arr.forEach((el) => {
      let coorX = el.substr(0, 1);
      arrCoordinates.push(coorX);
    });
    return this.isBingo(arrCoordinates);
  },
  checkDiagonalPattern(arr) {
    const arrStripped = Utils.eliminateDuplicates(arr);
    let isBingo = false;
    const arrPatternLeft = ["11", "22", "44", "55"];
    const arrPatternRight = ["15", "24", "42", "51"];
    isBingo = arrPatternLeft.every(elem => arrStripped.indexOf(elem) > -1);
    if (!isBingo)
      isBingo = arrPatternRight.every(elem => arrStripped.indexOf(elem) > -1);
    return isBingo;
  },
  checkCornersPattern(arr) {

```

```

const arrStripped = Utils.eliminateDuplicates(arr);
let isBingo = false;
const arrPattern = ["11", "15", "51", "55"];

isBingo = arrPattern.every(elem => arrStripped.indexOf(elem) > -1);

return isBingo;
};

export { WinningPatterns };

```

oo) winning/flying-prize.js

```

import { EventsConsts } from '../events/events-consts';
import Animator from '../utils/animator';
import ApiController from '../API/api-controller';

class FlyingPrize {
  constructor(sum) {
    this.sum = sum;

    document.addEventListener(EventsConsts.PRIZE_WON, () => {
      if (this.sum > 0) {
        FlyingPrize.animatePrizeFlying(this.sum);
        this.sum = 0;
      }
    });
  }

  static animatePrizeFlying(sum) {
    const elFlyingPrize = document.createElement('div');
    elFlyingPrize.setAttribute('id', 'flyingPrize');
    elFlyingPrize.innerHTML = sum;
    document.body.appendChild(elFlyingPrize);

    // Animate moving the element to given top and left positions
    Animator.moveDiagonally(elFlyingPrize, 45, 45, -45, 42, Animator.quad,
    3000, '%');

    document.addEventListener(EventsConsts.FLYING_PRIZE_ANIMATION_ENDS, () => {
      if (document.body.contains(elFlyingPrize)) {
        document.body.removeChild(elFlyingPrize);

        // Add the new prize sum to the player balance
        ApiController.setNewBalance(sum, false);
      }
    });
  }
}

export default FlyingPrize;

```

pp) winning/win-paterns-amin-module.js

```

import ViewManipulator from '../utils/view-manipulator';
import { EventsConsts } from '../events/events-consts';

```

```

class WinPatternsAnimModule {
  constructor(elem, rows, cols, pattern) {
    this.elem = elem;
    this.rows = rows;
    this.cols = cols;
    this.pattern = pattern;

    document.addEventListener(EventsConsts.START_GAME, () => {
      ViewManipulator.toggleVisibility(elem.parentElement.parentElement, true);
    });

    document.addEventListener(EventsConsts.END_GAME, () => {
      ViewManipulator.toggleVisibility(elem.parentElement.parentElement, false);
    });
  }

  // Create the DOM element
  createDomElement() {
    this.createDomElement();
  }

  createDomElement() {
    const elTable = document.createElement('table');
    let i = 1;
    while (i <= this.rows) {
      const elRow = document.createElement('tr');
      let j = 1;
      while (j <= this.cols) {
        const elCell = document.createElement('td');
        elCell.setAttribute('id', 'x'+j + 'y' + i);
        elRow.appendChild(elCell);
        j++;
      }
      elTable.appendChild(elRow);
      i++;
    }
    this.elem.appendChild(elTable);
  }

  this.startAnimation();
}

/**
 * Define which pattern animation to start
 */
startAnimation() {
  switch (this.pattern) {
    case 'horizontal':
      this.startHorizontalAnim();
      break;
    case 'vertical':
      this.startVerticalAnim();
      break;
    case 'diagonal':
      this.startDiagonalAnim();
      break;
  }
}

/**
 * Clear all cells background
 */

```

```

clearTable() {
  const arrCells = this.elem.querySelectorAll('td');
  let cellIdx = 0;
  while (cellIdx < arrCells.length) {
    if (arrCells[cellIdx].classList.contains('highlighted')) {
      arrCells[cellIdx].classList.remove('highlighted');
    }
    cellIdx++;
  }
}

startHorizontalAnim() {
  let yIdx = 1;
  setInterval(() => {
    this.clearTable();

    let xIdx = 1;
    while (xIdx <= this.cols) {
      const elCell = this.elem.querySelector(`#${x${xIdx}}y${yIdx}`);
      elCell.classList.add('highlighted');
      xIdx++;
    }

    if (yIdx === this.rows) {
      yIdx = 0;
    }

    yIdx++;
  }, 1000)
}

startVerticalAnim() {
  let xIdx = 1;
  setInterval(() => {
    this.clearTable();

    let yIdx = 1;
    while (yIdx <= this.rows) {
      const elCell = this.elem.querySelector(`#${x${xIdx}}y${yIdx}`);
      elCell.classList.add('highlighted');
      yIdx++;
    }

    if (xIdx === this.rows) {
      xIdx = 0;
    }

    xIdx++;
  }, 1000)
}

startDiagonalAnim() {
  let count = 1;
  setInterval(() => {
    this.clearTable();

    switch (count) {
      case 1:
        let leftDiagonalIdx = 1;
        while (leftDiagonalIdx <= this.rows) {
          const elCell = this.elem.querySelector(`#${x${leftDiagonalIdx}}y${leftDiagonalIdx}`);
          elCell.classList.add('highlighted');
          leftDiagonalIdx++;
        }
    }
  }, 1000)
}

```

```

const elCell =
this.elem.querySelector(`#${leftDiagonalIdx}y${leftDiagonalIdx}`);
elCell.classList.add('highlighted');
leftDiagonalIdx++;
}
break;
case 2:
let rightDiagonalIdx = 1;
while (rightDiagonalIdx <= this.rows) {
const elCell = this.elem.querySelector(`#${this.rows + 1 - rightDiagonalIdx}y${rightDiagonalIdx}`);
elCell.classList.add('highlighted');
rightDiagonalIdx++;
}
break;
case 3:
let cornersIdx = 1;
const elTopLeft =
this.elem.querySelector(`#${cornersIdx}y${cornersIdx}`);
elTopLeft.classList.add('highlighted');

const elTopRight = this.elem.querySelector(`#${this.rows + 1 - cornersIdx}y${cornersIdx}`);
elTopRight.classList.add('highlighted');

const elBottomLeft = this.elem.querySelector(`#${cornersIdx}y${this.rows + 1 - cornersIdx}`);
elBottomLeft.classList.add('highlighted');

const elBottomRight = this.elem.querySelector(`#${this.rows + 1 - cornersIdx}y${this.rows + 1 - cornersIdx}`);
elBottomRight.classList.add('highlighted');
break;
}
count++;
if (count > 3)
count = 1;
}, 1000);
}

export default WinPatternsAnimModule;

```

qq) winning/winning-dialog.js

```

import ApiController from '../API/api-controller';
import VanillaModal from 'vanilla-modal';
import { EventsConsts } from '../events/events-consts';
import FlyingPrize from './flying-prize';

class WinningDialog {
constructor(elementID) {
let bingos = 0;
this.elementID = elementID;
WinningDialog.attachListeners(bingos, elementID);
}

static attachListeners(bingos, elementID) {
document.addEventListener(EventsConsts.BINGO, () => {

```

```

bingos++;
});

document.addEventListener(EventsConsts.START_GAME, () => {
  bingos = 0;
});

document.addEventListener(EventsConsts.END_GAME, () => {
  setTimeout(() => {
    const objWinning = {
      elementID: elementID,
      bingos: bingos,
      elBingosContainer: null,
      elPrize: null,
      text: 'Bingos:'
    };
    ApiController.setBingoWins(bingos);
    WinningDialog.createDialog(objWinning);
  });
});

document.addEventListener(EventsConsts.NOT_ENOUGH_BALANCE, () => {
  const objWinning = {
    elementID: elementID,
    text: 'Not enough money. Please deposit.'
  };
  WinningDialog.createMsgDialog(objWinning);
});

static getHeaderImgClass(bingos) {
  // Define which header image to show
  if (bingos === 0) {
    return 'no-bingo';
  }

  if (bingos === 0 && ApiController.getPlayerBalanceFromStorage() === 0) {
    return 'no-bingo-no-money';
  }

  if (bingos === 1) {
    return 'winner-one-bingo';
  }

  if (bingos === 2) {
    return 'winner-two-bingos';
  }

  if (bingos > 2) {
    return 'winner-more-than-two-bingos';
  }
}

static createMsgDialog(objWinning) {
  const elDialog = document.querySelector(objWinning.elementID);
  const elHeader = document.querySelector('header');
  elHeader.classList.add('no-bingo-no-money');
  const elDialogContent = elDialog.querySelector('#content');
  elDialogContent.innerHTML = objWinning.text;
}

```

```

WinningDialog.openDialog(objWinning.elementID);
}

static createDialog(objWinning) {
const elDialog = document.querySelector(objWinning.elementID);
const elDialogContent = elDialog.querySelector('#content');
const elHeader = document.querySelector('header');

// Clear header classes
elHeader.classList = '';

if (!document.body.contains(document.querySelector('#bingos'))) {
  elDialogContent.innerHTML = '<div class="col-sm-4 col-xs-4">$objWinning.text</div>';
  objWinning.elBingosContainer = document.createElement('div');
  objWinning.elBingosContainer.setAttribute('id', 'bingos');
  objWinning.elBingosContainer.setAttribute('class', 'col-sm-3 col-xs-4');
  elDialogContent.appendChild(objWinning.elBingosContainer);
} else {
  objWinning.elBingosContainer = document.querySelector('#bingos');
  objWinning.elBingosContainer.innerHTML = '';
}

if (!document.body.contains(document.querySelector('#prize'))) {
  objWinning.elPrize = document.createElement('div');
  objWinning.elPrize.setAttribute('id', 'prize');
  objWinning.elPrize.setAttribute('class', 'col-sm-5 col-xs-6');
  elDialogContent.appendChild(objWinning.elPrize);
} else {
  objWinning.elPrize = document.querySelector('#prize');
}

elHeader.classList.add(WinningDialog.getHeaderImgClass(objWinning.bingos));

const prizeSum = objWinning.bingos * 50;
objWinning.elPrize.innerHTML = `${objWinning.bingos} x 50 = ${prizeSum}`;

while(objWinning.bingos > 0) {
  objWinning.elBingosContainer.innerHTML += '<span>' +
    '' +
    ' x 50' +
    '</span>';
  objWinning.bingos--;
}

const flyingPrize = new FlyingPrize(prizeSum);

WinningDialog.openDialog(objWinning.elementID);
}

static openDialog(elementId) {
  const modal = new VanillaModal({onClose:
WinningDialog.onCloseWinningModal});
  modal.open(elementId);
}

static onCloseWinningModal() {

```

```

// Dispatch new event when the dialog is closed
const event = new CustomEvent(EventsConsts.PRIZE_WON, {
  detail: {
    time: new Date()
  }, bubbles: true, cancelable: true
});
document.dispatchEvent(event);
}

export default WinningDialog;

```

rr) app.js

```

/***
 * Created by Mihail on 8/15/2016 - Started the project on St. Mary day
 */

import fetch from 'node-fetch';
import ApiConsts from './API/api-consts';
import Initializer from './initializer/initializer';

class App {
  static async start() {
    fetch(ApiConsts.CONF).then((response) => {
      if (response.status >= 400) {
        throw new Error("Bad response from server");
      }
      return response.json();
    }).then((config) => {
      Initializer.applyConfigurations(config);
    }).catch(error => console.log('>>> error fetching: ', error));
  }
}

export default App;

((App) => {
  App.start().then(console.log('Game Started!'));
})(App);

```

5. styles

a) back-office/components/_actions.scss

```

#actions {
  span {
    border-radius: 150%;
    border: 2px solid black;
    color: black;
    cursor: pointer;
    display: inline-block;
    font-weight: bold;
    height: 25px;
    margin: 5px 15px;
    padding-top: 3px;
    text-align: center;
  }
}

```

```

transition-duration: 0.1s;
transition-property: transform;
width: 33px;
&:hover {
  transform: scale(1.1);
}
}

.actions-info {
  border-color: #3498DB;
  color: #3498DB;
}

.actions-delete {
  border-color: #B7477D;
  color: #B7477D;
}

```

b) ***back-office/components/_back-office-.scss***

```

#backOffice {
  .toolbar {
    margin: 40px;
  }

  .toolbar-add {
    float: left;
  }

  .toolbar-search {
    float: right;
    input {
      font-size: $backOfficeInputsFontSize;
      padding: 10px;
    }
  }

  .toolbar-add-btn {
    font-size: 32px;
  }

  .back-office-datagrid {
    clear: both;
    margin: 40px;
    padding-top: 20px;
  }
}

```

c) ***back-office/components/_button.scss***

```

.button {
  border-radius: 28px;
  border: 0;
  box-shadow: 0 1px 1px #d9d9d9;
  color: #fff;
  cursor: pointer;
  display: inline-block;
}

```

```

font-size: 18px;
font-weight: bold;
padding: 5px 15px;
text-decoration: none;
transition-duration: 0.1s;
transition-property: transform;
@include lineaVerticalGradient($backOfficeHeaderColour, purple);

&:hover {
  transform: scale(1.1);
}

```

d) **back-office/components/_dialog.scss**

```

.dialog {
  border: 1px solid lightgray;

  .dialog-header {
    background-color: $backOfficeHeaderColour;
    color: $mainWheatColour;
    font-family: $mainFontFamily;
    font-size: 42px;
    margin: 0;
    padding: 10px 10px 10px 20px;
    text-align: left;
  }

  .dialog-body {
    border-bottom: 1px solid lightgray;
    border-top: 1px solid lightgray;
    max-height: 50vh;
    overflow-x: hidden;
    padding: 10px 30px 30px 30px;
    text-align: center;
    font-size: 18px;
  }

  .dialog-footer {
    padding: 10px;
    text-align: right;

    .button {
      margin: 10px;
    }
  }
}

.dialog-modal-open {
  overflow: hidden; /* no body scroll */
}

.dialog-modal {
  background: rgba(0, 0, 0, 0.7);
  bottom: 0;
  left: 0;
  position: fixed;
  right: 0;
  top: 0;
}

```

```
z-index: 1;

.dialog-modal-wrap {
  background: white;
  margin: 5% auto;
  width: 350px;
}

.dialog-dismiss {
  color: gray;
  cursor: pointer;
  text-decoration: underline;
}
```

e) back-office/components/form.scss

```
#form {
  .form-label {
    color: #A0A0A0;
    display: block;
    font-size: 12px;
    padding: 12px 0 6px;
  }

  .form-row {
    padding-left: 20px;
    text-align: left;

    input {
      font-size: $backOfficeInputsFontSize;
      padding: 10px;
    }
  }
}
```

f) back-office/components/_logo.scss

```
#logo {
  background: $fifthColColour url('../../../../../images/logo.png');
  background-size: cover;
  display: inline-block;
  height: 60px;
  vertical-align: middle;
  width: 100px;
}
```

g) back-office/components/_logout.scss

```
.logout-module {
  display: inline-block;
  float: right;
  margin-right: 40px;
}
```

h) Back-office/components/_table.scss

```
#main-table {
  table {
    width: 100%;
    display: table;
    border-collapse: separate;
    border-spacing: 2px;
    border-color: grey;
  }

  th {
    cursor: pointer;
    padding: 5px;
  }

  td {
    cursor: cell;
    padding: 5px;
    background-color: $backOfficeTableBackgroundColour;
  }
}

.toolbar {
  margin-left: 20px;

  a, button {
    background: #3498db linear-gradient(to bottom, #3498db, #2980b9);
    border-radius: 28px;
    box-shadow: 0 1px 3px #666666;
    color: #ffffff;
    font-size: 14px;
    padding: 10px 20px;
    text-decoration: none;
    border: 0;
    margin-right: 5px;
  }

  a:hover, button:hover {
    background: #3cb0fd linear-gradient(to bottom, #3cb0fd, #3498db);
    text-decoration: none;
  }
}
```

i) back-office/back-office-app.scss

```
@import "../fonts";
@import "../variables";
@import "../mixins";
@import "components/actions";
@import "components/button";
@import "components/dialog";
@import "components/form";
@import "components/logo";
@import "components/table";
@import "components/logout";
@import "components/back-office";

html, body {
```

```

background: $backOfficeBackgroundColour;
background-size: cover;
margin: 0;
padding: 0;
height: 100%;
font-family: $mainFontFamily;
}

.app-header {
background-color: $fifthColColour;
padding: 10px 5px;
color: $mainWheatColour;
font-size: 1.6em;
}

.login-form {
width: 350px;
margin: 100px auto;
text-align: center;

button {
float: right;
margin-top: 21px;
}

input {
width: 307px;
}
}

footer {
margin: 26px auto 0 auto;
font-size: 10px;
text-align: center;
color: lightslategray;
}

#errorMsg {
width: 300px;
margin: -33px auto 32px auto;
text-align: center;
color: red;
}

```

j) cards/_card.scss

```

/* set +1px because of the cell borders,
this was each cell is 50px,
271px div -> 250px table -> 50px cell */
#card {
text-align: center;
padding: 10px;
width: auto;
margin: .2em;
font-family: $mainFontFamily;
font-weight: bold;
font-size: 2em;
position: relative;
background-color: #fff;

```

```
box-shadow: inset 0 0 7px rgba(0,0,0,0.9);

&.bingo {
  background: url('../.../images/logo.png') no-repeat 5px 90px;
  table {
    td { display: none; }
  }
}

table {
  width: 100%;
  background-color: $mainWheatColour;
  color: black;
  box-shadow: 0 0 7px rgba(0,0,0,0.9);

  th, td {
    border: 1px solid lightgoldenrodyellow;
    text-shadow: 0 1px 1px white;
    width: 40px;
    height: 40px;
  }
}

th {
  color: white;
  text-align: center;

  &.firstCol {
    background-color: $firstColColour;
  }

  &.secondCol {
    background-color: $secondColColour;
  }

  &.thirdCol {
    background-color: $thirdColColour;
  }

  &.fourthCol {
    background-color: $fourthColColour;
  }

  &.fifthCol {
    background-color: $fifthColColour;
  }
}

td {
  position: relative;
  padding-left: 3px;
  padding-right: 3px;
  cursor: pointer;

  &.marked:before {
    content: "";
    position: absolute;
    top: 1%;
    left: 0;
    width: 100%;
    height: 100%;
  }
}
```

```

    border-radius: 50%;
    background: radial-gradient(circle at 50% 0px, #ffffff, rgba(255,
255, 255, 0) 58%);
    -webkit-filter: blur(5px);
    z-index: 2;
}

&.marked {
    background: radial-gradient(circle at 50% 120%, #e17000, #f6c079
10%, #e17000 80%, #e17000 100%);
    border-radius: 50%;
}

&.drawn {
    background: radial-gradient(circle at 50% 120%, #dce1af, #f6c079
10%, #e11b3c 80%, #93abe1 100%);
    border-radius: 50%;
}
}
}
}
}

```

k) dauber/_ball.scss

```

#ball {
    display: inline-block;
    margin: 0 auto;
    width: 40px;
    height: 40px;
    border-radius: 50%;
    border: 1px solid black;

    &:before {
        content: "";
        position: absolute;
        top: 1%;
        left: 0;
        width: 100%;
        height: 100%;
        border-radius: 50%;
        background: radial-gradient(circle at 50% 0px, #ffffff, rgba(255, 255,
255, 0) 58%);
        -webkit-filter: blur(5px);
        z-index: 2;
    }
}

#innerCircle {
    width: 65%;
    height: 65%;
    border-radius: 50%;
    background-color: white;
    margin: 0 auto;
    position: relative;
    top: 13%;

    span{
        display: block;
        width: 25px;
        margin: 0 auto;
    }
}

```

```

font-family: $mainFontFamily;
font-size: 1.3em;
font-weight: bold;
position: relative;
text-align: center;
color: black;
}

&.original_ballB {
  background: radial-gradient(circle at 50% 120%, #008ed6, #76deef 10%,
#055194 80%, #062745 100%);
}

&.original_ballI {
  background: radial-gradient(circle at 50% 120%, #83d100, #c3ec81 10%,
#83d100 80%, #83d100 100%);
}

&.original_ballN {
  background: radial-gradient(circle at 50% 120%, #e17000, #f6c079 10%,
#e17000 80%, #e17000 100%);
}

&.original_ballG {
  background: radial-gradient(circle at 50% 120%, #a71906, #e35a45 10%,
#a71906 80%, #a71906 100%);
}

&.original_ballO {
  background: radial-gradient(circle at 50% 120%, #642e88, #8f4ad2 10%,
#642e88 80%, #642e88 100%);
}

&.redish_ballB {
  background: radial-gradient(circle at 50% 120%, #008ed6, #ff0000 10%,
#055194 80%, #062745 100%);
}

&.redish_ballI {
  background: radial-gradient(circle at 50% 120%, #83d100, #ff0000 10%,
#83d100 80%, #83d100 100%);
}

&.redish_ballN {
  background: radial-gradient(circle at 50% 120%, #e17000, #ff0000 10%,
#e17000 80%, #e17000 100%);
}

&.redish_ballG {
  background: radial-gradient(circle at 50% 120%, #a71906, #ff0000 10%,
#a71906 80%, #a71906 100%);
}

&.redish_ballO {
  background: radial-gradient(circle at 50% 120%, #642e88, #ff0000 10%,
#642e88 80%, #642e88 100%);
}

```

l) dauber/_blower.scss

```
#blower {  
  display: none;  
  width: 330px;  
  height: 278px;  
  margin-bottom: 10px;  
  position: relative;  
  order: 2;  
  
#tube {  
  background: url("../images/blower_with_tube_transparent.png") 0  
  0;  
  position: relative;  
  width: 260px;  
  height: 75px;  
  margin-right: 65px;  
  
#elVisibleBallsContainer {  
  height: 47px;  
  width: 264px;  
  position: absolute;  
  margin-left: 15%;  
}  
  
#ball {  
  display: inline-block;  
  position: absolute;  
  top: 18%; // start position for ball animations  
  left: 78%; // start position for ball animations  
}  
  
canvas {  
  background: url("../images/blower_with_tube_transparent.png") -  
  122px -70px;  
  display: none;  
  border: 3px solid rgba(0, 0, 0, 0.5);  
  border-top: none;  
  -webkit-border-radius: 50%;  
  -moz-border-radius: 50%;  
  border-radius: 50%;  
  position: absolute;  
  top: 25%;  
  left: 36%;  
}
```

m) market-place/_market-space.scss

```
#marketPlace {  
  margin-top: 15px;  
}
```

n) market-place/_market-cards.scss

```
#marketPlaceCards {  
  label {  
    font-size: 4em;  
    cursor: pointer;  
  }  
  
  .container-fluid {  
    .col-sm-3 {  
      border-top: 5px solid white;  
      border-left: 5px solid white;  
      &:nth-child(4) {  
        border-right: 5px solid white;  
      }  
  
      &:hover {  
        @include selected-state(#555ce1);  
      }  
  
      &.selected {  
        @include selected-state(#555ce1);  
      }  
  
      label {  
        margin: -10px auto 0 auto;  
        width: 100%;  
        height: 100%;  
        cursor: pointer;  
      }  
  
      .cards {  
        input[type="radio"] {  
          width: 30px;  
          height: 30px;  
          cursor: pointer;  
        }  
  
        img {  
          height: 150px;  
          margin: 0 auto;  
        }  
  
        .price {  
          width: 60px;  
          float: right;  
          font-size: .5em;  
          text-align: right;  
        }  
  
        .price-icon {  
          background: url('../images/coin_25x25.png') no-repeat;  
          background-size: 25px;  
          width: 25px;  
          height: 25px;  
          display: inline-block;  
        }  
      }  
    }  
  }  
}
```

```
&:last-child {  
    .col-sm-3 {  
        border-bottom: 5px solid white;  
    }  
}  
}  
}
```

o) **winning/_flying-prize.scss**

```
#flyingPrize {  
    position: absolute;  
    left: 45%;  
    top: 45%;  
    color: $mainFrontColour;  
    font-size: 10em;  
    font-weight: bold;  
    @include win-color-effect();  
}
```

p) **winning/_winning-dialog.scss**

```
/* Modal Dialog default styles */  
.modal-links {  
    display: inline-block;  
    background: #fff;  
    padding: 40px;  
    box-shadow: 0 2px 0 rgba(0,0,0,.1);  
}  
  
.modal-links ul {  
    padding: 0;  
    margin: 0;  
    list-style: none;  
}  
  
.modal-links li {  
    background: #dfdfdf;  
}  
  
.modal-links a {  
    display: block;  
    background: #fff;  
    padding: 20px 30px;  
    color: #777;  
    text-transform: uppercase;  
    text-decoration: none;  
    font-size: 12px;  
    transition: 0.2s;  
}  
  
.modal-links a:hover {  
    transform: translate(-2px, -2px);  
}  
  
/**  
 * Modal starts here.  
*/
```

```
/*
.modal {
  display: none;
}

.vanilla-modal .modal {
  display: block;
  position: fixed;
  content: "";
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background: rgba(0, 0, 0, 0.6);
  z-index: -1;
  opacity: 0;
  transition: opacity 0.2s, z-index 0s 0.2s;
  text-align: center;
  overflow: hidden;
  overflow-y: auto;
  white-space: nowrap;
  -webkit-overflow-scrolling: touch;
}

.vanilla-modal .modal > * {
  display: inline-block;
  white-space: normal;
  vertical-align: middle;
  text-align: left;
}

.vanilla-modal .modal:before {
  display: inline-block;
  overflow: hidden;
  width: 0;
  height: 100%;
  vertical-align: middle;
  content: "";
}

.vanilla-modal.modal-visible .modal {
  z-index: 99;
  opacity: 1;
  transition: opacity 0.2s;
}

.modal-inner {
  position: relative;
  max-width: 90%;
  max-height: 90%;
  border-radius: 10px;
  overflow-x: hidden;
  overflow-y: auto;
  background: #fff;
  z-index: -1;
  opacity: 0;
  transform: scale(0);
  transition: opacity 0.2s, transform 0.2s, z-index 0s 0.2s;
}
```

```

.modal-visible .modal-inner {
  z-index: 100;
  opacity: 1;
  transform: scale(1);
  transition: opacity 0.2s, transform 0.2s;
}

.a.modal-close {
  position: absolute;
  z-index: 2;
  right: 0;
  top: 0;
  width: 25px;
  height: 25px;
  line-height: 25px;
  font-size: 13px;
  cursor: pointer;
  text-align: center;
  background: #fff;
  box-shadow: -1px 1px 2px rgba(0,0,0,0.2);
  &:hover {
    text-decoration: none;
  }
}

/* Winning Dialog styles */
#winningDialog {
  width: 500px;
  height: auto;
  font-family: $mainFontFamily;

  header {
    height: 120px;
    border-bottom: 1px solid $winningDialogMainColour;

    &.winner-one-bingo {
      background: url('../images/winner_one_bingo.png') no-repeat center;
      background-size: auto 100px;
    }

    &.winner-two-bingos {
      background: url('../images/winner_two_bingos.png') no-repeat center;
      background-size: auto 100px;
    }

    &.winner-more-than-two-bingos {
      background: url('../images/winner_more_than_two_bingos.png') no-repeat center;
      background-size: auto 100px;
    }

    &.no-bingo {
      background: url('../images/no_bingo.png') no-repeat center;
      background-size: auto 100px;
    }

    &.no-bingo-no-money {
      background: url('../images/no_bingo_no_money.png') no-repeat
    }
  }
}

```

```

center;
  background-size: auto 100px;
}

@include breakpointMaxWidth('xs') {
  background-position: left;
}
}

#content {
height: auto;
color: $winningDialogMainColour;
font-size: $mainWinningDialogFontSize;
border-bottom: 1px solid $winningDialogMainColour;

div:first-child {
  font-size: $mainWinningDialogFontSize + .4;
}

@include breakpointMaxWidth('xs') {
  div:nth-child(2) {
    font-size: .8em;
  }
}

div:last-child {
  font-size: $mainWinningDialogFontSize + .1;
  margin-top: 2%;
}

span {
display: flex;
img {
  margin-right: 5px;

  &.bingo-img {
    position: absolute;
    left: 17%;
    width: 40px;
    height: 40px;
    margin-top: 8px;
  }

  &.background-rotating-img {
    -webkit-animation:spin 4s linear infinite;
    -moz-animation:spin 4s linear infinite;
    animation:spin 4s linear infinite;

    @-moz-keyframes spin { 100% { -moz-transform: rotate(360deg); } }

    @-webkit-keyframes spin { 100% { -webkit-transform: rotate(360deg); } }

    @keyframes spin { 100% { -webkit-transform: rotate(360deg);
    transform:rotate(360deg); } }
  }
}
}

footer {

```

```

color: $winningDialogMainColour;
button {
  margin-top: 5px;
  margin-bottom: 5px;
  width: 100px;

  @include breakpointMaxWidth('xs') {
    margin-left: 5px;
  }
}
}
}

```

q) **winning/_winning-patterns-animation-module.scss**

```

#winPatternsAnimModule {
  border: $mainFrontColour;

  #container {
    height: 278px;
    width: 330px;
    text-align: center;

    #horizontal {
      display: inline-block;
      width: 48%;
      height: 48%;

      table {
        width: 100%;
        height: 100%;
        th, td {
          border: 1px solid $mainFrontColour;
        }
      }
    }

    #vertical {
      display: inline-block;
      width: 48%;
      height: 48%;

      table {
        width: 100%;
        height: 100%;
        th, td {
          border: 1px solid $mainFrontColour;
        }
      }
    }

    #diagonal {
      display: inline-block;
      width: 48%;
      height: 48%;

      table {
        width: 100%;
        height: 100%;
      }
    }
  }
}

```

```

th, td {
    border: 1px solid $mainFrontColour;
}

.highlighted {
    background-color: $mainWheatColour;
    @include win-color-effect();
}

```

r) _breakpoints

```

@mixin breakpointMinWidth($point) {
//1024px - laptop
@if $point == xxl {
    @media (min-width: 64em) { @content; }
}
//960px
@else if $point == l {
    @media (min-width: 60em) { @content; }
}
//768px - tablet
@else if $point == m {
    @media (min-width: 48em) { @content; }
}
//480px - mobile
@else if $point == xs {
    @media (min-width: 30em) { @content; }
}

@mixin breakpointMaxWidth($point) {
//1024px - laptop
@if $point == xxl {
    @media (max-width: 64em) { @content; }
}
//960px
@else if $point == l {
    @media (max-width: 60em) { @content; }
}
//768px - tablet
@else if $point == m {
    @media (max-width: 48em) { @content; }
}
//480px - mobile
@else if $point == xs {
    @media (max-width: 30em) { @content; }
}

```

s) _fonts.scss

```
// Google Fonts
@import
```

```
url(http://fonts.googleapis.com/css?family=Roboto+Slab|Open+Sans:400italic,  
700italic,400,700);
```

t) _footer.scss

```
#footer {  
  .main-btn {  
    background-color: $mainHeaderColour;  
    display: block;  
    margin: 0 auto;  
  }  
  
  #startBtn {  
    margin-bottom: 10px;  
  }  
}
```

u) _header.scss

```
#header {  
  height: 130px;  
  font-family: $mainFontFamily;  
  color: $mainFrontColour;  
  background: $mainHeaderColour url('../images/logo.png') no-repeat 1%  
60%;  
  
  @include breakpointMaxWidth('xs') {  
    background-size: 115px 100px;  
  }  
  
  h1 {  
    display: none;  
    height: 130px;  
    width: 500px;  
    margin-left: 48%;  
    line-height: 130px;  
    text-transform: uppercase;  
  }  
  
  @include breakpointMinWidth('l') {  
    display: block;  
  }  
  
  @include breakpointMaxWidth('xxl') {  
    font-size: 2em;  
  }  
}  
  
#userProfile {  
  h2 {  
    font-size: 1.6em;  
  }  
  
  @include breakpointMaxWidth('m') {  
    h2 {  
      font-size: .9em;  
    }  
  }  
}
```

```
h4 {  
    font-size: .8em;  
}  
}  
}  
}
```

v) _helpers.scss

```
.no-right-padding {  
    padding-right: 0 !important;  
}
```

w) _mixins.scss

```
@mixin selected-state($colour) {  
    -moz-box-shadow: inset 10px 10px 150px $colour;  
    -webkit-box-shadow: inset 10px 10px 150px $colour;  
    box-shadow: inset 10px 10px 150px $colour;  
}  
  
@mixin win-color-effect() {  
    text-shadow: 0 1px 0 #ccc,  
    0 2px 0 #c9c9c9,  
    0 3px 0 #bbb,  
    0 4px 0 #b9b9b9,  
    0 5px 0 #aaa,  
    0 6px 1px rgba(0,0,0,.1),  
    0 0 5px rgba(0,0,0,.1),  
    0 1px 3px rgba(0,0,0,.3),  
    0 3px 5px rgba(0,0,0,.2),  
    0 5px 10px rgba(0,0,0,.25),  
    0 10px 10px rgba(0,0,0,.2),  
    0 20px 20px rgba(0,0,0,.15);  
}  
  
@mixin lineaVerticalGradient($colour1, $colour2) {  
    background: $colour1;  
    background: -webkit-linear-gradient($colour1, $colour2); /* For Safari 5.1 to 6.0 */  
    background: -o-linear-gradient($colour1, $colour2); /* For Opera 11.1 to 12.0 */  
    background: -moz-linear-gradient($colour1, $colour2); /* For Firefox 3.6 to 15 */  
    background: linear-gradient($colour1, $colour2); /* Standard syntax */  
}
```

x) _register-page.scss

```
#registerPage {  
    width: 100%;  
    margin: 0 auto;  
    position: relative;  
    top: 50%;  
    transform: translateY(-50%);  
  
    @include breakpointMinWidth('xxl') {
```

```

    width: 30%;
}

.brand-text {
  font-family: $mainFontFamily;
}

.panel {
  box-shadow: none;
}

.form-control {
  display: block;
  width: 100%;
  height: 70px;
  padding: 33px 12px 0;
  font-size: 1.3em;
  line-height: 1.42857;
  color: #555555;
  background-color: #fff;
  border-radius: 0;
  border-bottom: 1px solid #ccc;
  border-top: none;
  border-left: none;
  border-right: none;
  box-shadow: none;
}

.sign-in-link {
  margin-top: 3%;
}

.social {
  ul {
    list-style: none;
    margin: 0;
    padding: 0;

    li {
      display: inline-block;
      width: 7%;
      a {
        color: white;
        &:hover {
          color: #286090;
        }
      }
    }
  }
}

```

y) _variables.scss

```

$mainFontFamily: 'Roboto Slab', serif;
$mainFrontColour: #ffffdd;
$mainWheatColour: #F5DEB2;

$firstColColour: #008ed6;

```

```

$secondColColour: #83d100;
$thirdColColour: #e17000;
$fourthColColour: #a71906;
$fifthColColour: #642e88;

$winningDialogMainColour: #e17010;
$mainWinningDialogFontSize: 1.2em;

$mainHeaderColour: #1a0850;

// Back office app
$backOfficeHeaderColour: #642e87;
$backOfficeBackgroundColour: #f0f0f0;
$backOfficeTableBackgroundColour: #fff;
$backOfficeInputsFontSize: 16px;

```

z) app.scss

```

@import '../../node_modules/bootstrap-sass/assets/stylesheets/bootstrap';
@import 'back-office/back-office-app';
@import '_fonts';
@import '_variables';
@import '_mixins';
@import '_breakpoints';
@import '_helpers';
@import '_header';
@import '_footer';
@import '_register_page';
@import 'market-place/_market-space';
@import 'market-place/_market_cards';
@import 'dauber/_blower';
@import 'dauber/_ball';
@import 'cards/_card';
@import 'timer/_timer';
@import 'winning/_winning-dialog';
@import 'winning/_flying-prize';
@import 'winning/_winning-patterns-animation-module';

html, body {
  height: 100%;
  margin: 0;
  padding: 0;
  background: url('../../images/background.jpg');
  -webkit-background-size: cover;
  background-size: cover;
}

h1 {
  margin: 0;
  font-size: 2em;
}

#gameWrapper {
  color: $mainFrontColour;

  #gameContainer {
    width: 90%;
    margin: 2% auto 0 auto;
    display: flex;
  }
}

```

```

align-content: flex-start;

@include breakpointMaxWidth('xs') {
  flex-wrap: wrap-reverse;
}

#leftGameScreen {
  order: 1;
  padding: 0;
  margin: 0;
  list-style: none;
}

#rightGameScreen {
  order: 2;
}
}

```

6. Main directory files

a) .babelrc

```
{
  "presets": [
    "@babel/preset-env",
    "@babel/preset-react",
    "@babel/preset-flow"
  ],
  "plugins": ["@babel/plugin-transform-runtime"]
}
```

b) .env.example

```
DB_SECRET=<yourdatabasepassword>
DB_URI=mongodb://127.0.0.1/<your database name>
```

c) .eslintrc

```
{
  "parser": "babel-eslint",
  "rules": {
    "max-len": [1, 120, 2, {"ignoreComments": true}]
  },
  "parserOptions": {
    "ecmaVersion": 6,
    "sourceType": "module",
    "ecmaFeatures": {
      "modules": true
    }
  }
}
```

d) .flowconfig

```
[ignore]
.* /react/node_modules/*
.* /fbemitter/node_modules/*

[include]
node_modules/classnames
node_modules/fbemitter
node_modules/imutable
node_modules/invariant
node_modules/react
node_modules/react-addons-test-utils
node_modules/react-dom

[libs]

[options]
esproposal.class_static_fields=enable
```

e) .gitignore

```
node_modules
build/
.idea
.env
```

f) back-office.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Bingo Bigul Back Office</title>
  <link rel="apple-touch-icon" sizes="57x57" href="/favicon/apple-icon-57x57.png">
  <link rel="apple-touch-icon" sizes="60x60" href="/favicon/apple-icon-60x60.png">
  <link rel="apple-touch-icon" sizes="72x72" href="/favicon/apple-icon-72x72.png">
  <link rel="apple-touch-icon" sizes="76x76" href="/favicon/apple-icon-76x76.png">
  <link rel="apple-touch-icon" sizes="114x114" href="/favicon/apple-icon-114x114.png">
  <link rel="apple-touch-icon" sizes="120x120" href="/favicon/apple-icon-120x120.png">
  <link rel="apple-touch-icon" sizes="144x144" href="/favicon/apple-icon-144x144.png">
  <link rel="apple-touch-icon" sizes="152x152" href="/favicon/apple-icon-152x152.png">
  <link rel="apple-touch-icon" sizes="180x180" href="/favicon/apple-icon-180x180.png">
  <link rel="icon" type="image/png" sizes="192x192" href="/favicon/android-icon-192x192.png">
  <link rel="icon" type="image/png" sizes="32x32" href="/favicon/favicon-32x32.png">
  <link rel="icon" type="image/png" sizes="96x96" href="/favicon/favicon-96x96.png">
```

```

<link rel="icon" type="image/png" sizes="16x16" href="/favicon/favicon-16x16.png">
<link rel="manifest" href="/favicon/manifest.json">
<meta name="msapplication-TileColor" content="#ffffff">
<meta name="msapplication-TileImage" content="/ms-icon-144x144.png">
<meta name="theme-color" content="#ffffff">
<link rel="stylesheet" type="text/css" href="build/styles/back-office.css">
</head>
<body>
<div id="backOfficeApp"></div>
<div class="alert alert-danger alert-dismissible" role="alert" id="errorMsg" style="display: none;">
<span id="messageText">Wrong login details.</span>
</div>
<script src="build/js/back-office.js"></script>
</body>
</html>

```

g) config.json

```

{
  "gameConf": {
    "id": "1",
    "name": "American Bingo",
    "numbers": [
      1, 2, 3, 4, 5,
      6, 7, 8, 9, 10,
      11, 12, 13, 14, 15,
      16, 17, 18, 19, 20,
      21, 22, 23, 24, 25,
      26, 27, 28, 29, 30,
      31, 32, 33, 34, 35,
      36, 37, 38, 39, 40,
      41, 42, 43, 44, 45,
      46, 47, 48, 49, 50,
      51, 52, 53, 54, 55,
      56, 57, 58, 59, 60,
      61, 62, 63, 64, 65,
      66, 67, 68, 69, 70,
      71, 72, 73, 74, 75
    ],
    "skin": {
      "name": "original"
    },
    "appTitle": "Welcome to Bingo Bigul",
    "turnsCount": 23,
    "freeSpotImgPath": "<img src='../../images/small_logo_30x30.png' />",
    "drawIntervalSeconds": 6,
    "beforeStartGameSeconds": 3,
    "marketCards": true,
    "dauber": true,
    "playingCards": true,
    "mainGame": true,
    "winningDialog": true,
    "cardPrice": 2,
    "winPatternsAnimModule": true
  }
}

```

h) discoverer.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Discoverer Tool</title>
  <link rel="apple-touch-icon" sizes="57x57" href="/favicon/apple-icon-57x57.png">
  <link rel="apple-touch-icon" sizes="60x60" href="/favicon/apple-icon-60x60.png">
  <link rel="apple-touch-icon" sizes="72x72" href="/favicon/apple-icon-72x72.png">
  <link rel="apple-touch-icon" sizes="76x76" href="/favicon/apple-icon-76x76.png">
  <link rel="apple-touch-icon" sizes="114x114" href="/favicon/apple-icon-114x114.png">
  <link rel="apple-touch-icon" sizes="120x120" href="/favicon/apple-icon-120x120.png">
  <link rel="apple-touch-icon" sizes="144x144" href="/favicon/apple-icon-144x144.png">
  <link rel="apple-touch-icon" sizes="152x152" href="/favicon/apple-icon-152x152.png">
  <link rel="apple-touch-icon" sizes="180x180" href="/favicon/apple-icon-180x180.png">
  <link rel="icon" type="image/png" sizes="192x192" href="/favicon/android-icon-192x192.png">
  <link rel="icon" type="image/png" sizes="32x32" href="/favicon/favicon-32x32.png">
  <link rel="icon" type="image/png" sizes="96x96" href="/favicon/favicon-96x96.png">
  <link rel="icon" type="image/png" sizes="16x16" href="/favicon/favicon-16x16.png">
  <link rel="manifest" href="/favicon/manifest.json">
  <meta name="msapplication-TileColor" content="#ffffff">
  <meta name="msapplication-TileImage" content="/ms-icon-144x144.png">
  <meta name="theme-color" content="#ffffff">
  <link type="text/css" href="build/styles/back-office.css" rel="stylesheet">
</head>
<body>
<div id="discoverer"></div>
<script src="build/js/back-office-discoverer.js"></script>
</body>
</html>
```

i) gulpfile.js

```
require("@babel/polyfill");
const gulp = require('gulp');
const sass = require('gulp-sass');
const eslint = require('gulp-eslint');
const concat = require('gulp-concat');
const del = require('del');
const postcss = require('gulp-postcss');
const autoprefixer = require('autoprefixer');
const rename = require('gulp-rename');
const server = require('gulp-server-livereload');
const sourcemaps = require('gulp-sourcemaps');
const source = require('vinyl-source-stream');
```

```

const buffer = require('vinyl-buffer');
const browserify = require('browserify');
const watchify = require('watchify');
const babelify = require('babelify');
const istanbul = require('gulp-istanbul');
const gulpJest = require('gulp-jest').default;

const paths = {
  html: 'index.html',
  scripts: './src/**/*.js',
  sass: './styles/**/*.scss',
  tests: './front-end-tests/_tests_',
  backOfficeScripts: './src/back-office/**/*.js',
  discovererScripts: './src/back-office/*.js',
  backOfficeSass: './styles/sass/back-office/**/*.scss',
  buildSass: './build/styles',
  buildScripts: './build/js',
  backOfficeTests: './back-office-tests/_tests_'
};

const clean = () => del(['build', 'coverage']);

const lint = (done) => {
  gulp.src(paths.scripts)
    .pipe(eslint())
    .pipe(eslint.format());
  done();
};

const scripts = (done) => {
  const bundler = watchify(browserify('./src/app.js', { debug: true })
    .transform(babelify));

  bundler.bundle()
    .on('[gulpfile] Error in scripts task:', (err) => {
      console.error(err);
      this.emit('end');
    })
    .pipe(source('app.js'))
    .pipe(buffer())
    .pipe(sourcemaps.init({ loadMaps: true }))
    .pipe(sourcemaps.write('.'))
    .pipe(gulp.dest(paths.buildScripts));
  done();
};

const scriptsBackOffice = (done) => {
  const bundler = watchify(browserify('./src/back-office/back-office-app.js',
    { debug: true })
    .transform('babelify', { presets: ["@babel/preset-env", "@babel/preset-react"] }));

  bundler.bundle()
    .on('[gulpfile] Error in scriptsBackOffice task', (err) => {
      console.error(err);
      this.emit('end');
    })
    .pipe(source('back-office.js'))
    .pipe(buffer())
};

```

```

    .pipe(sourcemaps.init({ loadMaps: true }))
    .pipe(sourcemaps.write('./'))
    .pipe(gulp.dest(paths.buildScripts));
    done();
};

const scriptsDiscoverer = () => {
  const bundler = watchify(browserify('./src/back-office/discoverer.js', {
    debug: true
  })
    .transform('babelify', { presets: ["@babel/preset-env", "@babel/preset-react"] }));
  bundler.bundle()
    .on('[gulpfile] Error in scriptsDiscoverer task', (err) => {
      console.error(err);
      this.emit('end');
    })
    .pipe(source('back-office-discoverer.js'))
    .pipe(buffer())
    .pipe(sourcemaps.init({ loadMaps: true }))
    .pipe(sourcemaps.write('./'))
    .pipe(gulp.dest(paths.buildScripts));
};

const styles = (done) => {
  gulp.src([paths.sass, '!./styles/sass/back-office/**/*.*scss'])
    .pipe(sourcemaps.init({ loadMaps: true }))
    .pipe(sass().on('error', sass.logError))
    .pipe(concat('all.css'))
    .pipe(postcss([autoprefixer({
      browsers: [
        'last 2 versions',
        'Android 4.4',
        'ie 10-11',
        'ios_saf 8'
      ]
    }]))
    .pipe(rename({ suffix: '.min' }))
    .pipe(sourcemaps.write('./'))
    .pipe(gulp.dest(paths.buildSass));
  done();
};

const stylesBackOffice = (done) => {
  gulp.src(paths.backOfficeSass)
    .pipe(sourcemaps.init({ loadMaps: true }))
    .pipe(sass().on('error', sass.logError))
    .pipe(concat('back-office.css'))
    .pipe(sourcemaps.write('./'))
    .pipe(gulp.dest(paths.buildSass));
  done();
};

const jestConfig = {
  rootDir: paths.backOfficeTests
};
const testsBackOffice = (done) => {
  gulp.src(paths.backOfficeTests).pipe(gulpJest({
    "preprocessorIgnorePatterns": [
      "<rootDir>/build/", "<rootDir>/node_modules/"
  })
  .on('error', (err) => {
    console.error(`Error in ${err.message}`);
    done();
  })
  .on('end', () => {
    done();
  })
});

```

```

],
"roots": ["back-office-tests"],
"automock": false
});
done();
};

const test = (done) => {
gulp.src(paths.tests).pipe(gulpJest({
"preprocessorIgnorePatterns": [
"<rootDir>/build/", "<rootDir>/node_modules/"
],
"roots": ["front-end-tests"],
"automock": false
}))
.pipe(istanbul.writeReports({
dir: './coverage',
reporters: ['lcov'],
reportOpts: { dir: './coverage' }
}))
.pipe(istanbul.enforceThresholds({ thresholds: { global: 90 } }));
done();
};

const watchBackOfficeScripts = () => gulp.watch(paths.backOfficeScripts,
['scriptsBackOffice']);
const watchDiscovererScripts = () => gulp.watch(paths.discovererScripts,
['scriptsDiscoverer']);
const watchBackOfficeStyles = () => gulp.watch(paths.backOfficeSass,
['stylesBackOffice']);
const watchScripts = () => gulp.watch(paths.scripts, ['scripts']);
const watchStyles = () => gulp.watch(paths.sass, ['styles']);
const watchTests = () => gulp.watch(paths.tests, ['test']);
const watchJestTests = () => gulp.watch([jestConfig.rootDir +
"/!**!/!*.*.js"], ['jest']);

const watch = (done) => {
gulp.parallel(
watchBackOfficeScripts,
watchDiscovererScripts,
watchBackOfficeStyles,
watchScripts,
watchStyles,
watchTests,
watchJestTests
);
done();
};

const webserver = (done) => {
gulp.src('./')
.pipe(server({
livereload: false,
open: true
}));
done();
};

const build = gulp.series(clean, gulp.parallel(
lint,

```

```

scripts,
scriptsBackOffice,
scriptsDiscoverer,
styles,
stylesBackOffice,
testsBackOffice,
test,
watch,
webserver
));

const buildFrontEnd = gulp.series(clean, gulp.parallel(
  lint,
  scripts,
  styles,
  test,
  watch,
  webserver
));

const buildBackOffice = gulp.series(clean, gulp.parallel(
  scriptsBackOffice,
  scriptsDiscoverer,
  stylesBackOffice,
  testsBackOffice,
  watch,
  webserver
));

exports.default = build;
exports.fe = buildFrontEnd;
exports.bo = buildBackOffice;

```

j) index.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Welcome to Bingo Bigul</title>
  <link rel="apple-touch-icon" sizes="57x57" href="/favicon/apple-icon-57x57.png">
  <link rel="apple-touch-icon" sizes="60x60" href="/favicon/apple-icon-60x60.png">
  <link rel="apple-touch-icon" sizes="72x72" href="/favicon/apple-icon-72x72.png">
  <link rel="apple-touch-icon" sizes="76x76" href="/favicon/apple-icon-76x76.png">
  <link rel="apple-touch-icon" sizes="114x114" href="/favicon/apple-icon-114x114.png">
  <link rel="apple-touch-icon" sizes="120x120" href="/favicon/apple-icon-120x120.png">
  <link rel="apple-touch-icon" sizes="144x144" href="/favicon/apple-icon-144x144.png">
  <link rel="apple-touch-icon" sizes="152x152" href="/favicon/apple-icon-152x152.png">
  <link rel="apple-touch-icon" sizes="180x180" href="/favicon/apple-icon-180x180.png">

```

```

<link rel="icon" type="image/png" sizes="192x192" href="/favicon/android-icon-192x192.png">
<link rel="icon" type="image/png" sizes="32x32" href="/favicon/favicon-32x32.png">
<link rel="icon" type="image/png" sizes="96x96" href="/favicon/favicon-96x96.png">
<link rel="icon" type="image/png" sizes="16x16" href="/favicon/favicon-16x16.png">
<link rel="manifest" href="/favicon/manifest.json">
<meta name="msapplication-TileColor" content="#ffffff">
<meta name="msapplication-TileImage" content="/ms-icon-144x144.png">
<meta name="theme-color" content="#ffffff">
<link rel="stylesheet" href="build/styles/all.min.css"/>
<link href="//maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-awesome.min.css" rel="stylesheet">
</head>
<body>
<div class="vertical-align text-center" id="registerPage">
<div class="page-content vertical-align-middle">
<div class="panel">
<div class="panel-body">
<div class="brand">

<h2 class="brand-text font-size-18">Bingo Bigul</h2>
</div>
<!-- Register Form START -->
<form method="post" id="registerForm" accept-charset="utf-8" style="display: none;">
<div class="form-group form-material floating">
<input type="text" class="form-control" id="registerName" name="registerName" placeholder="Your name..." required/>
</div>
<div class="form-group form-material floating">
<input type="email" id="registerEmail" name="registerEmail" class="form-control" placeholder="Your email..." required/>
</div>
<div class="form-group form-material floating">
<input type="password" class="form-control" id="registerPassword" name="registerPassword" placeholder="Your password..." required/>
</div>
<button class="btn btn-primary btn-block btn-lg margin-top-40" type="submit" id="registerBtn" name="registerBtn">
Sign Up
</button>
</form>
<!-- Register Form END -->
<!-- Login Form START -->
<form method="post" id="loginForm" accept-charset="utf-8">
<div class="form-group form-material floating">
<input type="email" class="form-control" id="email" name="email" placeholder="Your email..." required/>
</div>
<div class="form-group form-material floating">
<input type="password" class="form-control" id="password" name="password" placeholder="Your password..." required/>
</div>

```

```

        </div>
        <button class="btn btn-primary btn-block btn-lg margin-top-40" type="submit" id="loginBtn" name="loginBtn">
            Login
        </button>
    </form>
<!-- Login Form END -->
<div class="sign-in-link">
    <a href="#"><span class="icon0-question-sign icon0-white"></span>
Don't have an account? Register here!</a>
</div>
<div class="alert alert-danger alert-dismissible" role="alert" id="alertMsg" style="display: none;">
    <button id="btnCloseAlertMsg" type="button" class="close" data-dismiss="alert" aria-label="Close">
        <span aria-hidden="true">&times;</span>
    </button>
    <span id="messageText">Wrong login details.</span>
</div>
</div>
<div class="social">
    <ul>
        <li><a href="https://github.com/mihailgaberov"><i class="fa fa-github"></i></a></li>
        <li><a href="https://es.linkedin.com/in/mihail-gaberov-6a73b03a"><i class="fa fa-linkedin"></i></a></li>
        <li><a href="https://twitter.com/mihailgaberov"><i class="fa fa-twitter"></i></a></li>
        <li><a href="https://www.facebook.com/profile.php?id=100000218793770"><i class="fa fa-facebook"></i></a></li>
    </ul>
</div>
</div>
</div>
<div id="gameWrapper" style="display: none;">
<section id="header">
    <div class="container-fluid">
        <div class="col col-sm-9">
            <h1>Welcome to Bingo Bigul</h1>
        </div>
        <div class="col col-sm-3 text-right">
            <section id="userProfile">
                <h2>Gosho Porcheto</h2>
                <div><a href="mailto: gogo22@gmail.com">gogo22@gmail.com</a></div>
                <h4>balance: <span>534</span></h4>
            </section>
        </div>
    </div>
</section>
<section id="marketPlace">
    <div id="marketPlaceCards">
        <div class="container-fluid">
            <div class="col-xs-6 col-sm-3 jsMarketCardContainer selected">
                <label for="one">
                    <div class="cards">
                        1<input type="radio" id="one" name="marketCards" value="1"

```

```

checked="checked">
    
    <div class="price">2</div>
</div>
</label>
</div>
<div class="col-xs-6 col-sm-3 jsMarketCardContainer">
    <label for="two">
        <div class="cards">
            2<input type="radio" id="two" name="marketCards" value="2">
            
            <div class="price">4</div>
        </div>
    </label>
</div>
<div class="col-xs-6 col-sm-3 jsMarketCardContainer">
    <label for="three">
        <div class="cards">
            3<input type="radio" id="three" name="marketCards" value="3">
            
            <div class="price">6</div>
        </div>
    </label>
</div>
<div class="col-xs-6 col-sm-3 jsMarketCardContainer">
    <label for="four">
        <div class="cards">
            4<input type="radio" id="four" name="marketCards" value="4">
            
            <div class="price">8</div>
        </div>
    </label>
</div>
<div class="container-fluid">
    <div class="col-xs-6 col-sm-3 jsMarketCardContainer">
        <label for="five">
            <div class="cards">
                5<input type="radio" id="five" name="marketCards" value="5">
                
                <div class="price">10</div>
            </div>
        </label>
    </div>
    <div class="col-xs-6 col-sm-3 jsMarketCardContainer">
        <label for="six">
            <div class="cards">
                6<input type="radio" id="six" name="marketCards" value="6">
                
                <div class="price">12</div>
            </div>
        </label>
    </div>
    <div class="col-xs-6 col-sm-3 jsMarketCardContainer">
        <label for="seven">
            <div class="cards">
                7<input type="radio" id="seven" name="marketCards" value="7">
                
                <div class="price">14</div>
            </div>
        </label>
    </div>

```

```

</label>
</div>
<div class="col-xs-6 col-sm-3 jsMarketCardContainer">
<label for="eight">
<div class="cards">
  8<input type="radio" id="eight" name="marketCards" value="8">
  
  <div class="price">16</div>
</div>
</label>
</div>
</div>
</div>
</section>
<div id="gameContainer">
<!-- Winning modal START-->
<div class="modal">
<div class="modal-inner">
  <a data-modal-close class='modal-close'>X</a>
  <div class="modal-content"></div>
</div>
</div>
<div id="winningDialogContainer" style="display: none">
<section id="winningDialog">
<header></header>
<div id="content" class="container-fluid">
</div>
<footer>
  <button data-modal-close class="modal-close btn btn-lg center-block">OK</button>
</footer>
</section>
</div>
<!-- Winning modal END-->
<div id="timerContainer">
  <p class="pulsate inner-shadow"></p>
</div>
<section id="rightGameScreen">
<section id="blower">
  <canvas id="blower-balloon" class="img-responsive"></canvas>
  <div id="tube" class="img-responsive"></div>
</section>
<section id="winPatternsAnimModule" style="display: none;">
<div id="container">
  <div id="horizontal"></div>
  <div id="vertical"></div>
  <div id="diagonal"></div>
  <div id="empty"></div>
</div>
</section>
</section>
<section id="leftGameScreen" class="container-fluid"></section>
</div>
<section id="footer">
<div class="container-fluid text-center">
  <button class="btn btn-primary btn-lg main-btn" id="startBtn">START GAME</button>
  <button class="btn btn-primary btn-lg main-btn" id="logoutBtn">LOGOUT</button>
</div>

```

```

</section>
</div>

<script src="build/js/app.js"></script>
</body>
</html>

```

k) package.json

```

{
  "name": "bingo",
  "version": "1.0.1",
  "description": "JavaScript based Bingo game developed using TDD approach unit testing with Mocha and Chai. Backend with React/Express/NodeJS and MongoDB for database.",
  "main": "src/app.js",
  "jest": {
    "transform": {
      "^.+\\.jsx?$": "babel-jest",
      "^.+\\.js?$": "babel-jest"
    },
    "setupFiles": [
      "./setupJest.js",
      "jest-canvas-mock"
    ]
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/mihailgaberov/bingo.git"
  },
  "keywords": [
    "javascript",
    "js",
    "react",
    "tdd",
    "bingo",
    "game"
  ],
  "author": "Mihail Gaberov",
  "license": "MIT",
  "dependencies": {
    "body-parser": "^1.18.2",
    "dotenv": "^2.0.0",
    "express-jwt": "^5.3.1",
    "jsonwebtoken": "^8.5.1",
    "node-fetch": "^2.6.0",
    "passport": "^0.3.2",
    "passport-local": "^1.0.0"
  },
  "devDependencies": {
    "@babel/core": "^7.1.6",
    "@babel/plugin-transform-runtime": "^7.1.0",
    "@babel/polyfill": "^7.0.0",
    "@babel/preset-env": "^7.7.6",
    "@babel/preset-flow": "^7.0.0",
    "@babel/preset-react": "^7.0.0",
    "@babel/register": "^7.0.0",
    "@babel/runtime": "^7.1.5",
    "@testing-library/react": "^9.4.0",
  }
}

```

```

"autoprefixer": "^6.7.7",
"babel-core": "^6.26.3",
"babel-eslint": "^9.0.0",
"babel-jest": "^23.6.0",
"babelify": "^10.0.0",
"bootstrap-sass": "^3.3.7",
"browserify": "^16.2.3",
"classnames": "^2.2.5",
"del": "^2.2.2",
"event-emitter-es6": "^1.1.5",
"express": "^4.16.3",
"fbemitter": "^2.1.1",
"gulp": "^4.0.2",
"gulp-babel": "^8.0.0",
"gulp-concat": "^2.6.0",
"gulp-eslint": "^6.0.0",
"gulp-istanbul": "^1.1.3",
"gulp-jest": "^4.0.3",
"gulp-mocha": "^3.0.1",
"gulp-postcss": "^6.4.0",
"gulp-rename": "^1.2.2",
"gulp-sass": "^4.0.2",
"gulp-server-livereload": "^1.8.4",
"gulp-sourcemaps": "^1.12.1",
"immutable": "^3.8.2",
"invariant": "^2.2.4",
"jest-canvas-mock": "^2.2.0",
"jest-cli": "^24.9.0",
"jest-fetch-mock": "^3.0.0",
"mongoose": "^5.7.14",
"paper": "^0.12.4",
"react": "^16.12.0",
"react-dom": "^16.12.0",
"vanilla-modal": "^1.6.5",
"vinyl-buffer": "^1.0.1",
"vinyl-source-stream": "^2.0.0",
"watchify": "^3.11.1"
},
"bugs": {
  "url": "https://github.com/mihailgaberov/bingo/issues"
},
"homepage": "https://github.com/mihailgaberov/bingo#readme"
}

```

I) README.md

Bingo Bigul
Famous Bingo game built with [JavaScript](<https://www.javascript.com/>). It has a back office app built with [ReactJS](<https://reactjs.org/>) and uses [MongoDB](<https://www.mongodb.com/>) for database.

```
![Main login screen](

217


```

```

-screenshots/game
![Game screen](https://github.com/mihailgaberov/bingo/blob/master/screenshots/game
-screenshots/back-
![Back office login screen](https://github.com/mihailgaberov/bingo/blob/master/screenshots/back-
office-login-screen.png)
![Back office screen](https://github.com/mihailgaberov/bingo/blob/master/screenshots/back-
office-screen.png)
![Back office edit screen](https://github.com/mihailgaberov/bingo/blob/master/screenshots/back-
office-edit-screen.png)
![Discoverer screen](https://github.com/mihailgaberov/bingo/blob/master/screenshots/discoverer-
screen.png)

## Running The App

```

To run the app, follow these steps:

1. Ensure that [NodeJS](http://nodejs.org/) is installed. This provides the platform on which the build tooling runs.
2. From the project folder, execute the following command:

```

```shell
npm install
```

```

3. Ensure that [Gulp](http://gulpjs.com/) is installed globally. If you need to install it, use the following command:

```

```shell
npm install -g gulp
```

```

> **Note:** Gulp must be installed globally, but a local version will also be installed to ensure a compatible version is used for the project.

4. Make sure you have ` `.env` file in your main project directory, containing the correct values for the following:

```

```dotenv
DB_SECRET=<yourdatabasepassword>
DB_URI=mongodb://127.0.0.1/<your database name>
```

```

5. Make sure you have [MongoDB](https://www.mongodb.com/download-
center/community) installed and you are able to use [MongoDB shell version v4.0.0](https://docs.mongodb.com/manual/mongo/index.html)

6. To start MongoDB server, execute the following command:

```

```
mongod
```

```

> The default `dbpath` for MongoDB is `/data/db`, which means that if you have installed MongoDB in your `C:\` drive `mongod` will use it implicitly. But if you have your database installed on different path, you need to use `--dbpath=/path/to/your/db` parameter to specify it. It could be for example like this: `mongod --dbpath d:\mongodb\data\db`. For more info on this you may

take a look [here](<https://docs.mongodb.com/manual/reference/configuration-options/>).

7. After you have your database running, you need to start the game server. You can do it by executing the following command in the project main directory:

```
```shell
node server.js
```
```

8. To run the app, execute the following command:

```
```shell
gulp
```
```

9. To run only the game client + unit tests, execute the following command:

```
```shell
gulp fe
```
```

10. To run only the back office app + unit test for it, execute the following command:

```
```shell
gulp bo
```
```

11. Browse to <http://localhost:8000> to see the app.

m) server.js

```
const express = require('express');
const bodyParser = require('body-parser');
const passport = require('passport');
const path = require('path');

require('dotenv').config();
require('./bingo-api/models/db');
require('./bingo-api/config/passport');

const routesApi = require('./bingo-api/routes/index');

const app = express();

const allowCrossDomain = function (req, res, next) {
  res.header('Access-Control-Allow-Origin', 'http://localhost:8000');
  res.header('Access-Control-Allow-Methods', 'GET, PUT, POST, DELETE, OPTIONS');
  res.header('Access-Control-Allow-Headers', 'Content-Type, authorization');
  next();
};

app.use(allowCrossDomain);
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
```

```
app.use(passport.initialize());
app.use('/bingo-api', routesApi);

app.use(function(req, res) {
  res.sendFile(path.join(__dirname, '/', 'index.html'));
});

app.use(function (err, req, res) {
  if (err.name === 'UnauthorizedError') {
    res.status(401);
    res.json({ "message" : err.name + ": " + err.message });
  }
});

app.listen(8888);
```

n) **setupJest.js**

```
require('jest-fetch-mock').enableMocks();
```