Пловдивски университет „Паисий Хилендарски”
Факултет по математика и информатика
Специалност Софтуерни технологии и дизайн

ДИПЛОМНА РАБОТА

Тема

Мултиплатформено приложение за агрегиране и филтриране на дигитална информация

Дипломант:
Васил Григоров Костадинов
Фак.No 1901681053

Научен ръководител:
гл. ас. д-р Иван Димитров

Пловдив, 2023

Съдържание

Списък с фигури

I. Увод

Интернет - едно от най-важните и значителни неща в съвременния свят, глобалната мрежа, свързваща всички дигитални и smart устройства на планетата. Тя представлява сложна съвкупност от възли, изградени чрез използване на физически връзки, като проводници и презокеански кабели или безжични предаватели като мобилни клетки и релета. Точно тези връзки дават възможност на всяко едно устройство да предава и приема информация от което и да е друго такова, позволявайки на хората да си комуникират на големи разстояния.

Създаването на глобалната мрежа е резултат от постоянното търсене на нови хоризонти за информационните технологии и жаждата за още мощност, скорост и пространство за съвременните устройства. Първата нейна версия е създадена през 60-те години като мрежова връзка между няколко университета в САЩ. С годините все повече университети започват да се включват към мрежата, претоварвайки я, което принуждава учените да премислят начина по който се пренасят данни. През 70-те години се започва разработката на пакетна комуникация - вид комуникация, при който потокът от данни се разбива на определена бройка битове с цел успешното им пристигане и се изпраща към финалната дестинация. При този метод, за разлика от директния поток на данни, каквато е например FM трансмисията, няма значение по кой път ще мине даден пакет с информация и кога ще пристигне, тъй като позицията му е закодирана в него. При получаване на всички данни, информацията се сглобява като пъзел. Това е така наречения TCP протокол.

През 80-те години тази версия на глобална мрежа се разраства още повече и се превръща в бледа версия на съвременната технология.

С всяка следваща година потенциалът на интернет расте и правопропорционално с него и развитието му. Все повече компании започват да виждат смисъла в една глобална мрежа, свързваща всички хора по света. Множество средства започват да се влагат в пускането му на пазара като технология със свободен достъп за широката общественост. Започват да се стандартизират протоколи и дигиталните устройства се произвеждат с портове и релета за интернет.

С времето все повече дейности започват да се изместват онлайн. Комуникацията се трансформира от физическа линия, направена от метал, през телефонни обаждания по радиовълни до свръхскоростни съобщения, съдържащи снимки, видеа, текст, гласови файлове и други със скорост над 1 гигабит в секунда. Източници на информация, възникнали като аналогови, като например новинарски емисии, прогнози за времето, календари с информация за лични събития, фотоалбуми или видеоленти, започват да се дигитализират и изместват своето съдържание на облака. Заради този облак, хората променят начина, по който могат да

се забавляват, да учат или да работят. Тоновете информация може с лекота да бъдат записани и след това достъпвани от всяко място по света. Благодарение на интернет, в момента почти всеки човек носи в джоба си устройство, съчетаващо в една малка обвивка, килограми аналогови устройства, като телевизор, калкулатор, тефтер, пишеща машина, телефон, фотоапарат, видеокамера, дистанционно управление, карти, компас, фенерче, вестник, радио, касетофон и много, много други.

Бъдещето на интернет изглежда още по-блестящо от настоящето. С навлизането на текстови модели, генератори на изображения чрез стабилна дифузия и други видове изкуствен интелект, които работят по висикоскоростната мрежа, доставяйки резултати почти мигновено, човек може да получи много повече и по-акуратна информация от когато и да е било. Модели с изкуствен интелект започват да се имплементират и в сферата на транспорта - монтират се модули за самоуправление в автомобили, камиони, влакове. С времето всяко едно превозно средство ще знае позицията на другите около него, което ще намали значително инцидентите. Софтуер за умно самоуправление се влага и в системите за контрол над трафика, уличното осветление, билборди и табели за навигация. Базово ниво на AI се инсталира дори и на домакински уреди като хладилници, печки, ходещи прахосмукачки и косачки.

Накратко, интернет позволява светкавичната обмяна на голям обем от информация на големи или малки разстояния и то във всяка сфера от съвременния живот, без изключения. Но с толкова много новопоявили се източници на информация, съществува вероятността някои от тях да предават дезинформация.

Целта на проекта е да се разработи цялостен софтуерен продукт, който да събира и филтрира информация от вече доверени източници, като я разпределя в определени категории и подкатегории и я обозначава с ниво на важност. Той също предоставя възможност за следене на любими източници и медии и дори създаване на собствен feed с новини. Продуктът се състои от следните компоненти:

- PWA (Progressive Web Application) приложение, което може да бъде инсталирано на всякакво устройство, без значение от операционната система, стига то да поддържа поне един от известните уеб браузъри.
- API сървър + администраторски панел, който да обработва заявки, да връща данни и да кешира страници. Админ панелът представлява визуална репрезентация на всички таблици в базата данни.
- Meilisearch индексираща машина и търсачка, която бързо да върне нужните резултати на потребителя, като филтрира милиони записи.

Продукти като Google News имплементират функционалността да показват новини и събития, базирани на интересите на потребителя, които са събрани, чрез следене на неговата активност, но те не винаги са точни. Те също не предоставят възможност за следене на дадени източници и изграждане на персонален feed.

Целта на софтуерния продукт е да удовлетворява нуждите на потребителите, които желаят да бъдат информирани за случващото се около тях, но желаят да следят само източниците, които те харесват, като цялата тази информация излиза на едно място и на тях не им се налага да търсят и да сменят много различни приложения и сайтове.

Втора цел на продукта е да даде поле за изява на млади колумнисти, които да документират събитията в тяхното обкръжение. Могат да се създават профили на квартали, улици или дори отделни входове на жилищни сгради, където могат да се публикуват новини, интересуващи само тази малка група от потребители.

За разработка на PWA приложението са използвани следните технологии: HTML, CSS, JavaScript, TypeScript, React, ReactDOM, NEXT.js, Node.js, Apollo GraphQL, Meilisearch.js

За разработка на API сървъра е използвана рамката Strapi.js, която пристига с готов интерфейс и генерация на заявки чрез Apollo GraphQL.

За разработката на Meilisearch индексиращата машина е използвана библиотеката Meilisearch.js, която е свързана директно с базата данни.

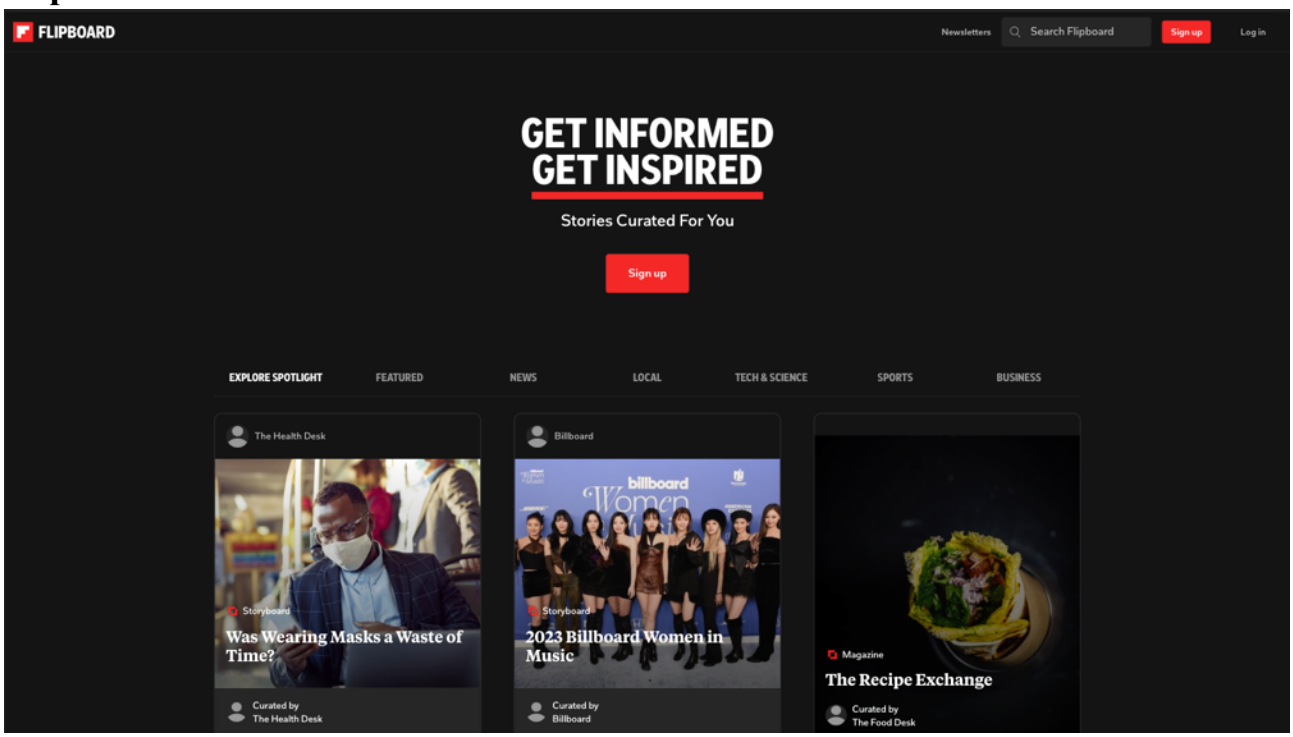
За база от данни е използван PostgreSQL.

За всички компоненти на продукта е използван един програмен език - TypeScript, който е разширение на JavaScript - езикът по подразбиране на всеки браузъри най-популярният скриптов език. Чрез използване на новонавлезлия стандарт PWA, JavaScript приложения могат да се компилират до native за конкретния OS.

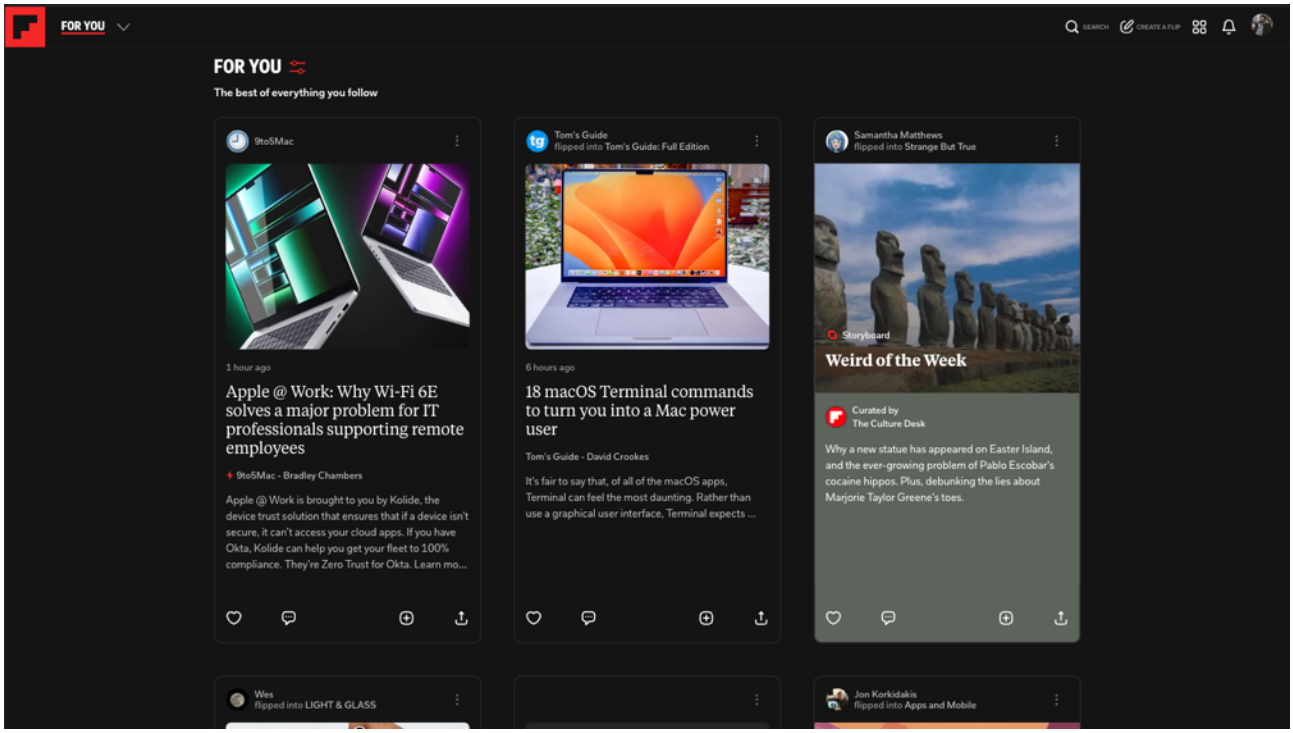
Структура на проекта. Проектът се състои от увод, 3 глави и заключение. Глава първа съдържа информация за няколко подобни приложения, глава втора представя технологиите, използвани за направата на мултиплатформеното приложение, а глава трета демонстрира функционалността.

II. Подобни приложения

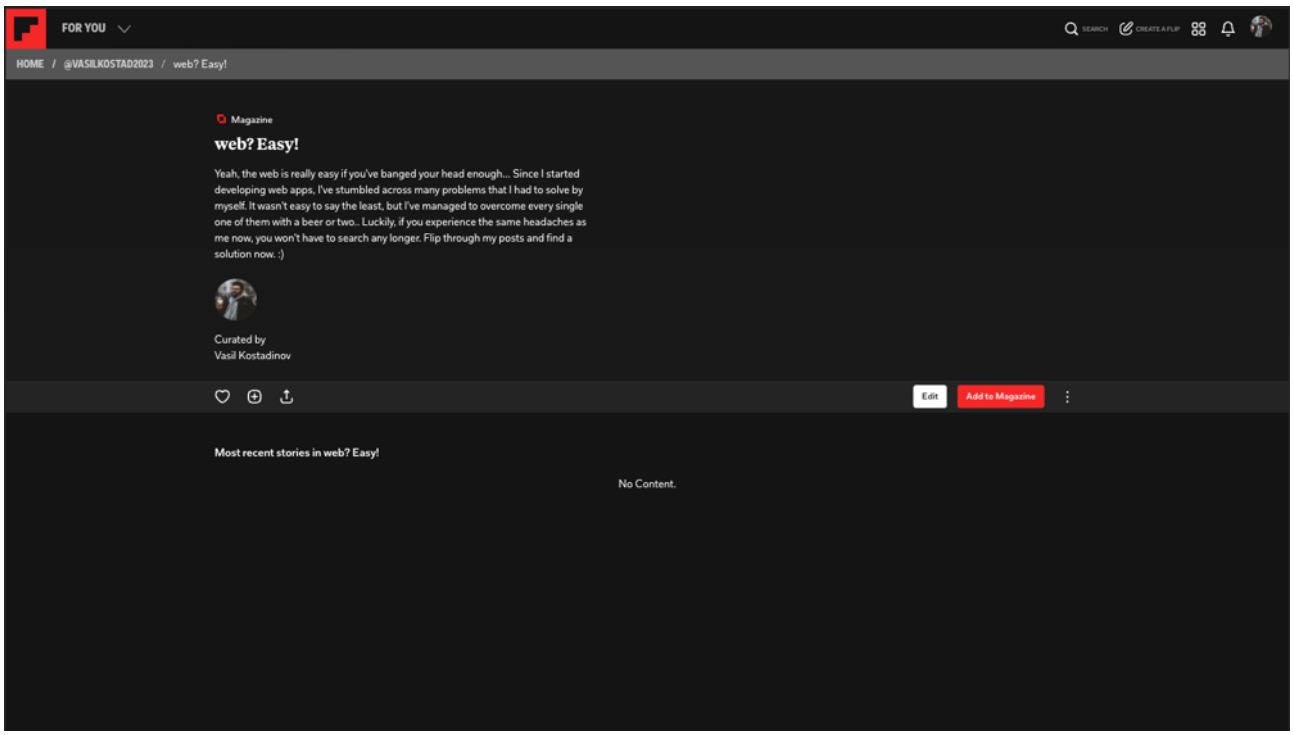
Flipboard



Фиг. 1 Flipboard - Агрегатор на новини и социални медии



Фиг. 1 Flipboard - Персонализиран Feed с истории

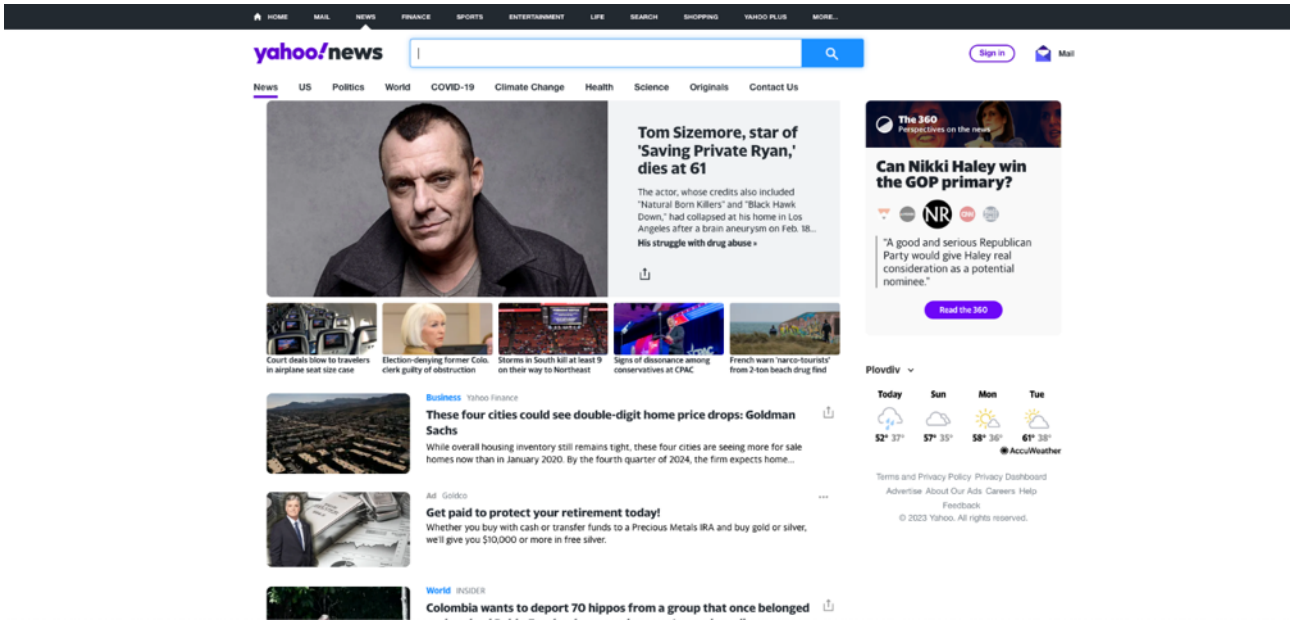


Фиг. 3 Flipboard - Възможност за създаване на собствени истории

Flipboard представлява React уеб и мобилно приложение за четене и споделяне на различни истории. Приложението притежава следните функционалности за:

1. Нерегистриран потребител
    - Възможност за четене на последните и/или най-популярни истории в предварително подбрани категории
    - Възможност за търсене на новини, колекции от новини (списания) по категории, тагове, автори, списания или теми
    - Възможност за абониране към Newsletter
    - Възможност за входиране или регистрация
2. Регистриран потребител
    - Всички правомощия на нерегистриран потребител
    - Възможност за персонализиране на публичен профил чрез добавяне на снимка, псевдоним и биография
    - Възможност за следване на теми, категории и потребители и изграждането на персонализирана начална страница с резултати
    - Възможност за създаване на собствена тема, собствено списание и собствени истории

- Табло с информация за представянето на всяка история, брой четения, брой харесвания, брой коментари (analytics)

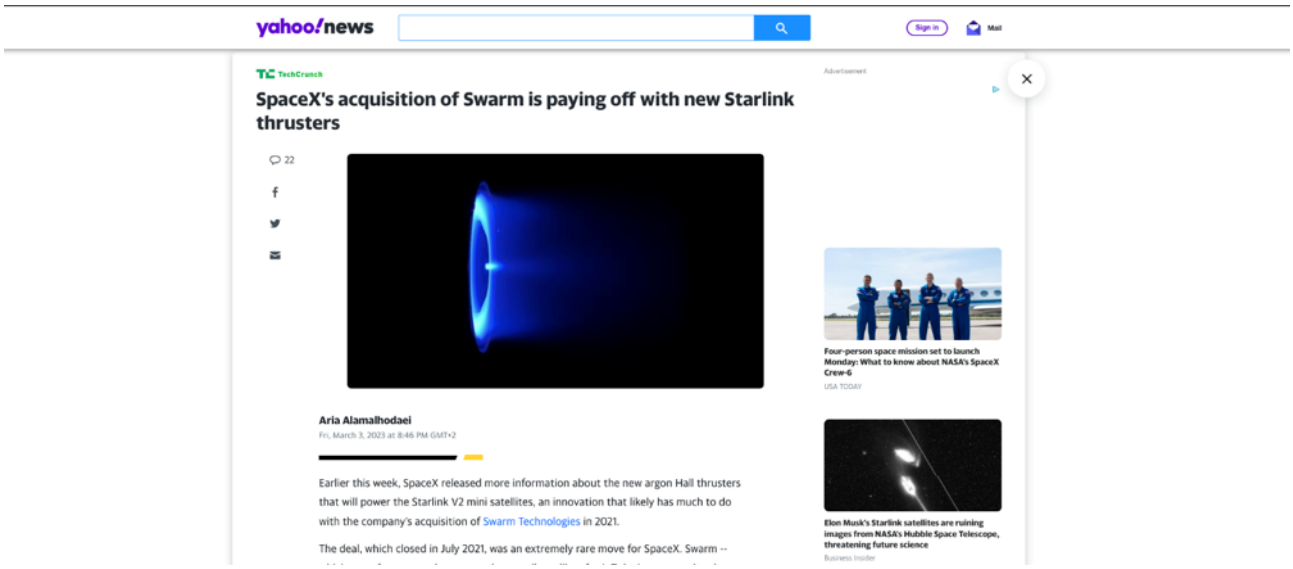


Фиг. 4 Yahoo! News. Сайт за новини и прогноза за времето.

Yahoo! news

React Уебсайт, предоставящ възможност за четене и споделяне на новини и оставяне на коментари под тях, както и за проверка на прогнозата за времето в текущия район.

Приложението предлага:



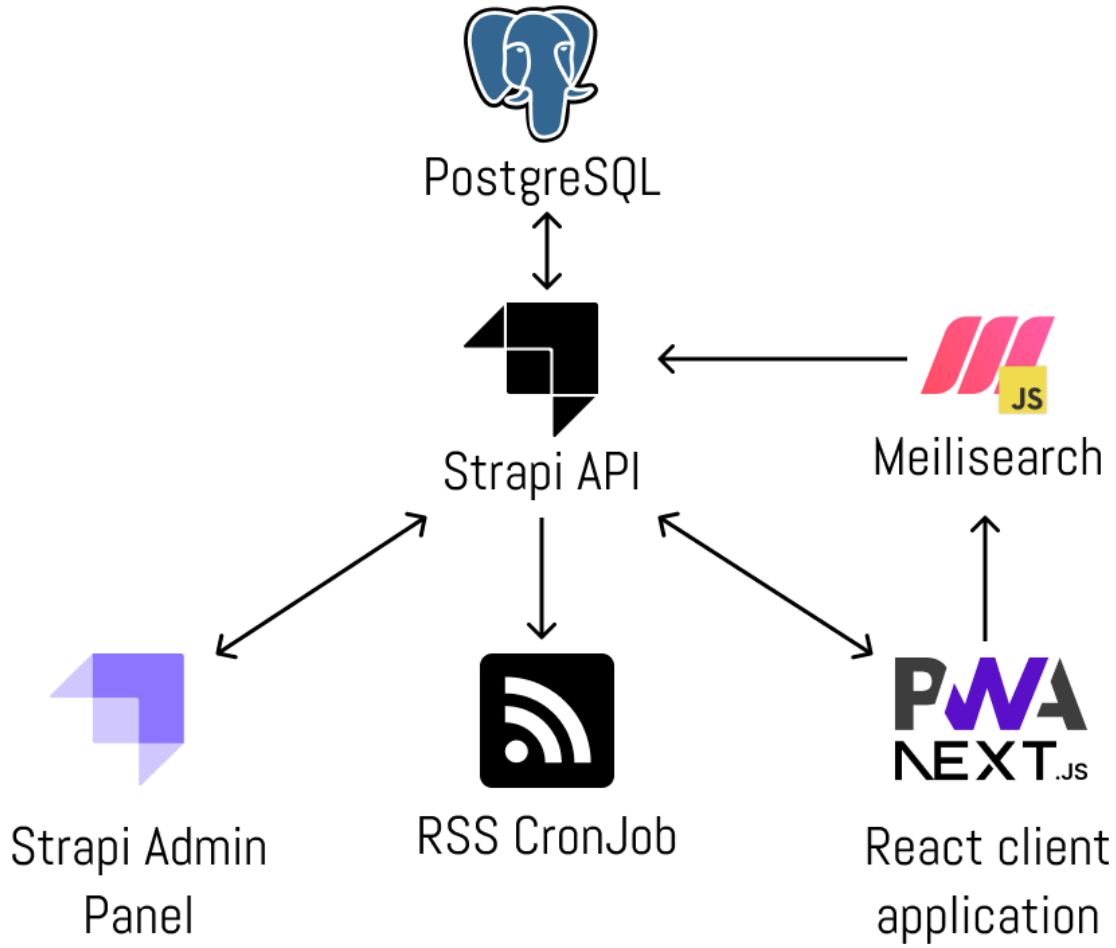
Фиг. 5 Yahoo! News. Детайлен преглед на новина.

- Детайлно търсене на новини, теми и уебсайтове, чрез търсещата машина на Yahoo!
- Филтриране на новини по категории и източници

- Възможност на рекламодатели да поставят реклами под/до най-четените новини
- Прогноза за времето за 4 дни напред, предоставена от AccuWeather
- Възможност за коментиране на новини и споделянето им в социални мрежи

III. Използвани средства, архитектура и технологии

Архитектура на приложението



Фиг. 6 Схема на приложението, администрацията, автоматизираните процеси и търсачката

Използвани технологии

За направата на приложението са използвани следните технологии и езици за програмиране: HTML, CSS, JavaScript, TypeScript, React.js, NEXT.js, GraphQL, Node.js, Strapi, PostgreSQL, MeiliSearch

HTML

HyperText Markup Language или по-познат като HTML (произнася се “ейч-ти-ем-ел”, в превод от английски език “език за маркиране на хипертекст”) е основният маркиращ език за изграждане на уеб страници, а вече и по-известните уеб приложения и прогресивни уеб приложения. Той позволява създаването на структурирани блокови елементи с текст, хиперлинкове, списъци и други HTML елементи, които съставят уеб страницата. HTML позволява създаването не само на уеб страници, а и на всякакъв друг вид текстови документи и може да се използва като текстов редактор, подобен на MS Word, Apple Pages и т.н.

HTML не се счита за програмен език, понеже с него не може да се изгради динамична функционалност или логика. Счита се обаче за стандартен и задължителен език при създаването на уеб съдържание.

Разработката на HTML започва през 1989 в ЦЕРН (на английски език CERN или European Organisation for Nuclear Research, превод от английски език “Европейска организация за ядрени изследвания”) от Тим Бърнърс-Лий (на английски език Tim Berners-Lee), създател на първия в света Уеб браузър “World Wide Web”. За работата на браузъра е бил нужен някакъв вид език за създаване и редактиране на съдържание, което може лесно да бъде форматирано от създателя, интерпретирано от браузъра и визуализирано на екрана на крайния потребител. Синтаксисът, на който Тим се спира, е базиран на SGML (“Standard Generalized Markup Language, превод от английски език “Стандартен обобщен език за маркиране”), който използва отварящи и затварящи тагове в триъгълни скоби, които обгръщат всяко парче текст, за да му придадат смисъл. Примери за такива тагове са: <h1/> за заглавие, <p/> за параграф, <a/> за хиперлинк към друга уеб страница или (division, от англ. дивизия, разделяне), който служи за визуално разделение на други специализирани тагове и за изграждане на комплексни интерфейси.

Езикът преминава през много различни версии с развитието на интернет технологиите от 1991 насам. Последната най-стабилна и вече стандартизирана версия излиза през 2008 година под името HTML5 и е версията, която всички използваме днес.

Модерната версия на HTML използва тагове за визуализация на различни ресурси като например: <video/> се използва за представянето на видеа, <audio/> - за изпълнение на звук, а <canvas/> - за изрисуването на векторни графики в реално време.

Типичният HTML елемент притежава отварящ и затварящ таг (<отварящ></затварящ>), като между тях се помещава съдържанието. То съдържа текст или друг/други HTML елементи, наричани още “деца” (от английски език “children”). Отварящият таг също може да притежава един или повече атрибути вътре в себе, които са способни да персонализират външния вид или поведението на елемента. Те работят на принципа “ключ-стойност”, като стойността се

изписва в кавички след знак равно: attribute=“value” и се позиционира преди втората триъгълна скоба на отварящият таг: Link.

За да се създаде един HTML документ, трябва първо да се създаде файл с разширение .html. Този файл съдържа йерархия от възли, базирани на DOM моделът (Document Object Model). Уеб документът стартира с <html></html> таг, който има точно две деца: <head></head> и <body></body>.

<head/> съдържа информация, която не се показва директно на потребителя като заглавието на документа или неговото описание. В <head/> частта се съдържат още и: мета информация за документа и връзки към други скриптови и стилови файлове.

<body/> съдържа конкретния потребителски интерфейс, който ще се показва на екрана на крайния потребител. Той може да започва с <h1>Заглавие<h1/>, следвано от параграф с информация .

CSS

Cascading Style Sheets (произнася се “Каскейдинг стайл шийтс”) или CSS (произнася се “си-ес-ес”, превод от английски език “Каскадни Стилови Таблици”) е стилов език, създаден за допълнително стилизиране и подобряване презентацията на един HTML документ, чрез модифициране на размера, цвета, позиционирането и оформлението на блоковите елементи.

Първата версия на CSS излиза през 1996г. и еволюира до CSS3 през 1999г. Нови функционалности към 3-та версия на езика продължават да се добавят и до ден днешен.

Езикът работи на принципа на правила. Типичното CSS правило започва със селектор (selector), който може да бъде името на някой HTML таг или стойността на някой от неговите атрибути, например class. Селекторът избира всеки един елемент в документа, който отговаря на условията. След селектора се инициализира декларативен блок чрез къдрави скоби, където се описват визуалните свойства на елемента.

Всеки един селектор може да се комбинира с друг за получаване на по-голяма специфичност на селектора и таргетиране на точно определен елемент. Съществуват и така наречените “псевдо селектори”, които таргетират точно определено състояние на един елемент, например ако той е застъпен от курсора (hover).

Особеността на CSS е, че всеки един стил или правило, което се дефинира е каскадно. Това означава, че могат да се създадат няколко правила за един и същи елемент и те ще бъдат комбинирани в едно, като ако има повтарящи се свойства в две или повече от правилата, ще бъде взета най-последната стойност или иначе казано тази, която е написана последна.

Правилата също притежават свойството за специфичност, което позволява на някои правила да имат по-голяма тежест върху други, според това къде са дефинирани и най-вече какъв селектор е използван.

В допълнение, някои елементи могат да наследяват стила на своите родители, докато други - не. Свойства като цвят и големина на текст могат да бъдат наследени от най-близкия родител.

В заключение, браузърът използва свойствата на правилата за каскадност, спицифичност и наследяване, за да контролира кои стилове се прилагат върху конкретен елемент.

Пример за CSS правило е:

```
div:odd h1:hover {
    color: #ff0;
    font-size: 15px;
}
```

Тук “div” и “h1” са селектори, “:odd” и “:hover” са псевдо селектори, а “color” и “font-size” са свойства. “#ff0” и “15px” са стойностите на съответните свойства. Логиката на това правило е следната: Търсят се всички заглавия <h1/>, които са застъпени от курсора и са деца на елемент, който сам по себе си е нечетно дете. След като бъдат открити в текущият документ, се задават техните свойства за цвят и размер.

CSS правила могат да се дефинират в HTML файл, като се отвори специален за тази цел таг <style/>. Те също могат да бъдат дефинирани в отделен файл с разширение .css, като след това пътят към този файл трябва да се посочи в <link/> тага в <head/> частта на конкретен HTML файл.

JavaScript

JavaScript (произнася се “Джаваскрипт”) е еднонишков, интерпретиран език за програмиране от високо ниво, известен с това, че се използва за изграждането на сложни уеб сайтове и уеб приложения.

Създаден е през 1995г. само за една седмица от Brendan Eich (произнася се Брендън Айх) с цел да бъде интегриран като скриптов език в браузъра Netscape (произнася се “Нетскейп”).

Първоначалното име на езика е било Mocha (произнася се “мока”), но след това бива променено на JavaScript с маркетингова цел, за да звучи подобно на новия по онова време обектно-ориентиран език за програмиране Java (произнася се “джава”). След това е стандартизиран под името EcmaScript (произнася се “екмаскрипт”).

В днешни времена езикът е известен с това, че е единственият освен Web Assembly (произнася се “уеб асембли”), който е вграден във всеки един браузър и се използва за

направата на уебсайтове. JavaScript може да се използва и за други цели като създаването на сървърни приложения чрез Node.js (произнася се “ноуд джей ес”), мобилни приложения чрез React Native (произнася се “реакт нейтив”) и desktop приложения чрез Electron (произнася се “електрон”).

Програмите на JavaScript са интерпретирани, а технологии като V8 Engine (произнася се “ви ейт енджин”) и Chromium (произнася се “хромиум”) използват Just-In-Time компилатори (от английски език “точно на време”), които ги превръщат в машинен код на момента. Прието е JavaScript програмите да се наричат “скриптове”.

Основните особености на езика са, че поддържа обектно ориентиран и функционален стил на програмиране и че е отличен в справянето с интензивни процеси, въпреки факта, че е еднонишков език. Това е възможно благодарение на така нареченият “неблокиращ цикъл от събития” (non-blocking event loop), който може да структурира и изпраща някои процеси за обработка във фонов режим, без да блокира основната нишка.

JavaScript скриптове могат да се дефинират в HTML файл, като се отвори специален за тази цел таг <script/>. Те също могат да бъдат дефинирани в отделен файл с разширение .js, като след това пътят към този файл трябва да се посочи в <script/> таг в <head/> частта на конкретен HTML файл.

JavaScript може да влияе на почти всяка част от браузъра, може да прихване всеки един елемент от HTML документа и да извлече неговите свойства: способен е да добавя, променя и премахва свойства на конкретен елемент. Умее дори да генерира нови .html, .css и .js файлове по програмен начин в сървърна среда, което позволява огромна пластичност на приложенията.

TypeScript

TypeScript (произнася се “тайпскрипт”) надгражда JavaScript като добавя типизиране на променливи, възможност за създаване и наследяване на интерфейси, подаване на тип към функция, дефиниране на върнат тип от фунцкия и много други функционалности, които позволяват по-лесното писане и поддръжка на код.

Тъй като TypeScript обгръща JavaScript в нови функционалности, всеки JavaScript код е валиден TypeScript код. След създаването на TypeScript кода, той се компилира до нормален JavaScript, като по време на компилацията се откриват всички грешки и неточности в типизирането, които могат да предизвикат срив в runtime-а (в изпълнението на приложението). След успешна компилация, всички типове биват премахнати от TypeScript файла, за да може да се разчете от browser-а.

За да се създаде тип в TypeScript е нужно да се напише запазената дума “type”, последвана от името на типа, знак за равно “=” и след това дефиницията на типа. Типът може

да бъде прост като string или number, или пък може да бъде сложен - обект, съдържащ properties, на които също може да се дефинират типове.

Пример за типове:

type SimpleType = string

type ComplexType = { name:string, id: number, vat: Array<string>, color:{ primary: string} }

Задаването на тип към променлива се извършва по следният начин: инициализира се променлива като се напише някоя от запазените думи: “const”, “let” или “var”. След това се изписва името на променливата, знак за две точки “:” и името на типа.

const myVariable: string = “hello”

const myOtherVariable: ComplexType = {name: “Ivan” , id:1, vat:[0,0,4,5], color: {primary: #fff }}

TypeScript може сам да задава типове по подразбиране (т.н infer type) на промеливите, базиран на тяхната стойност. Например променливата myVariable има стойност “hello”, която по подразбиране е string. В тези случаи не е нужно да се дефинира типа при инициализирането.

Синтаксисът за създаване на интерфейс е подобен на този за създаване на тип: изписва се запазената дума interface, последвана от името на интерфейса и къдрави скоби, където се дефинират свойствата му. Интерфейсите в TS, подобно на други езици, имат конвенция да се изписват с представка I (пр.: “IMyInterface”). Те също така могат да наследяват други интерфейси:

interface IPerson extends IHuman{ name: string }

Разликата между interface и type е, че interface-ът трябва да бъде статично зададен, трябва да започва с представка “I” и може да наследява други интерфейси, докато типът бива динамичен и извършва процесите обединение и сечение с други типове, но не може да ги наследява.

И двата синтаксиса дават възможност за опционални свойства, дефинирани със знак “?” след двоеточието:

type Optional = { name?: string, id: number}

където “name” не е задължително свойство.

В заключение, TypeScript предоставя възможност кода бъде тестван още докато бива писан и почти премахва дългият процес на debugging след написването му, което оптимизира доста работният процес.

React.js

ReactJS (или просто React, произнася се “Реакт”) е библиотека с отворен код (open-source) за разработване на потребителски интерфейси (UI) за уеб приложения и single-page апликации.

Нарича се single-page, защото всеки един екран от приложението се рендира на една HTML страница, като се заменя съдържанието и чрез процесът “пререндериране”. Кодът на React е отворен и достъпен за потребителите на официалната му страница в платформата “GitHub” и може да бъде свалян и модифициран спрямо нуждите на проекта.

Създаден е от Facebook през май 2013г. и се използва за създаване на компоненти, които могат да се преизползват, да се комбинират и да се организират в приложения.

Тъй като React базира синтаксиса си на HTML, може да се каже, че използва декларативен подход за създаване на компоненти, което означава, че разработчиците могат да опишат как трябва да изглежда интерфейсът, а React и browser-ът се грижат за това да се рендира коректно на потребителския екран.

Самият синтаксис на React са нарича JavaScript XML или накратко JSX, който позволява смесването на 3-те езика за front-end програмиране (HMTL, CSS и JavaScript) в един файл с разширение .jsx (.tsx ако използваме TypeScript).

Използва принципа на виртуален приложно-програмен интерфейс (“Virtual DOM”, произнася се “вирчуал дом”), който разширява функциите на нормалния DOM. DOM или Document Object Model представлява един голям обект с програмни инструкции, извлечени от HTML-а, за това как трябва да изглежда потребителският интерфейс. Бързодействието на React идва от това, че преди да направи каквото и да е пререндериране на екрана, той сравнява състоянието на виртуалния DOM преди и след промяната (дълбоко сравнение на два обекта), локализира точно какво е било променено и след това го подава към реалния DOM, който от своя страна пререндерира или “прерисува” само тази част от страницата на екрана. Без React, промяната в една част от документа ще се отрази в прерисуване на всички деца надолу по веригата, което прави процеса бавен и скъп от към производителност.

Виртуалният DOM вкарва още една концепция - тази за “състоянието” (state, произнася се “стейт”). Само специални променливи, маркирани като “променливи на състоянието” (state variables, произнася се “стейт вериабълс”), могат да предизвикат прерисуване на DOM-а. За тяхното изчисление могат да бъдат използвани и нормални променливи, но ако промяната на стойността им трябва да се отрази на екрана, то точно тази стойност трябва да бъде записана в състоятелна променлива.

Архитектурата на React позволява да се създават отделни парчета HTML код, които могат да се преизползват и комбинират на много места, което спестява време, пространство и ресурси. Така наречените “компоненти”. Компонентите могат да връщат:

- прост и основно презентационен HTML код
- динамичен код, зависещ от състоятелни променливи
- друг компонент, на който могат да бъдат подавани променливи за динамичност.

Те притежават състояние, което може да бъде модифицирано, използвайки състоятелните променливи и/или посредством параметрите, подадени към тях чрез родителския им компонент. Компонентите имат и жизнен цикъл, което означава, че се извикват специални методи при появата, съществуването и изчезването на даден компонент от DOM-а

Компонентите могат да бъдат дефинирани като клас или като функция във файл с разширение .jsx/.tsx. В най-новите версии на библиотеката няма значение кой от двата метода ще бъде използван, но за по-добър се смята функционалният, понеже е по лесен за писане и поддръжка.

Класовите компоненти се създават като се дефинира клас, който наследява абстрактният клас React.Component. След инициализирането му, в конструктора се извиква super конструкторът на наследения клас и се създава вътрешната променлива this.state, която ще отговаря за състоянието на компонента. Дефинира се и променливата this.props, която взема стойността си от конструктора и също се смята като променлива, променяща състоянието. Тя представлява параметрите, подадени към този компонент от неговия родител. JSX се връща от специален метод render().

Други специализирани методи са тези, които се извикват от жизнения цикъл на компонента или така наречените “lifecycle methods” (произнася се “лайфсайкъл методс”). Някои от тях са:

- componentDidMount() - извиква се при първата поява на компонента върху виртуалния ДОМ
- componentDidUpdate() - извиква се при всяка промяна на състоянието на компонента в this.state
- componentWillUnmout() - извиква се точно преди компонентът да изчезне от виртуалния ДОМ

- shouldComponentUpdate() - извиква се когато визуалното състояние на компонента не трябва се пререндира при промяна в this.state или props
- getSnapshotBeforeUpdate() - извиква се непосредствено преди промяна в състоянието и записва текущото състояние за референция по-късно

Функционалните компоненти се създават като се дефинира функция с един параметър props - обект със свойства, подадени от родителския компонент. Функцията, сама по себе си, връща JSX в нейният return.

След версия 16.8 на React е добавен контрол на състоянието и във функционалните компоненти чрез така наречените “Куки” или “Хукове” (hooks, поизнася се “хуукс”). Те винаги се изписват с представката “use” и трябва да се извикват на най-горно ниво в компонента. За инициализиране на състоятелна променлива се използва специалната кука useState(), като за параметър може да се подаде начална стойност на променливата. useState() връща масив с два елемента, първият от които е стойността на променливата, а вторият е метод за промяна на стойността ѝ, така наречения setter (произнася се “сетър”). Пример за useState:

```
const [value, setValue] = useState(0)
```

За контрол над жизнения цикъл се извиква друга специална кука useEffect, която приема за първи параметър функция (така наречената callback фунцкия), която да се изпълнява при някакво условие, а за втори параметър - масив с условия. Ако се подаде празен масив, то функцията ще се изпълни само при появата на компонента върху DOM-а. Ако пък се подаде масив с променливи, то функцията ще се изпълни при всяка промяна на елемент от масива. За да се извика функция при размонтирането или изчезването на компонента от DOM-а, то тази функция трябва да бъде върната от callback функцията на useEffect. Пример за useEffect:

```
useEffect(function lifeCycleFunction() {
  //Извивка се веднъж при монтиране
  calledOnArrayChange()

  return function calledOnUnmount(){
  // Извиква се при размонтиране
  }
}, [changedVariable]) //условия за извикване на функцията отново.
```

Примери за други куки са:

- useMemo - пази стойността на променлива след пререндериране и се използва за трудно и бавно изчислими променливи
- useCallback - пази стойността на цяла функция, за да не се предефинира след всяко пререндериране
- useRef - Пази стойност на променлива, която трябва да бъде запазена след пререндериране, но не трябва да се предизвиква пререндериране при промяна

В заключение React предоставя нов и различен метод за създаване на уеб приложения от конвенционалния. Синтаксисът наподобява и разширява класическия HTML, като позволява лесното вкарване и изобразяване на променливи, превръщайки кода в щампа, която може да бъде попълвана и преизползвана. Методите за жизнен цикъл позволяват по-висок контрол над състоянието и създаването на комплексни интерфейси със сложна логика. Целият проект и всичките му екрани се изрисуват на една HTML страница, като съдържанието се подменя от виртуалния DOM, чрез използването на състоятелни променливи.

NEXT.js

NEXT.js е framework (произнася се “фреймуърк” и означава рамка), който е специално разработен за създаване на уеб приложения, оптимизирани за производителност и SEO. Създаден е през 2016г. от частната компания Vercel (произнася се “Версъл”). Той е проектиран да опрости разработката на React приложения със сървърно рендериране, като прави създаването на бързи, динамични уеб страници лесно и бързо, с минимална конфигурация. NEXT.js предоставя редица функционалности, които помагат на разработчиците да създават уеб приложения, включително автоматично разделяне на кода, сървърно рендериране, генериране на статични страници и динамично маршрутизиране (“routing”, произнася се “раутинг”).

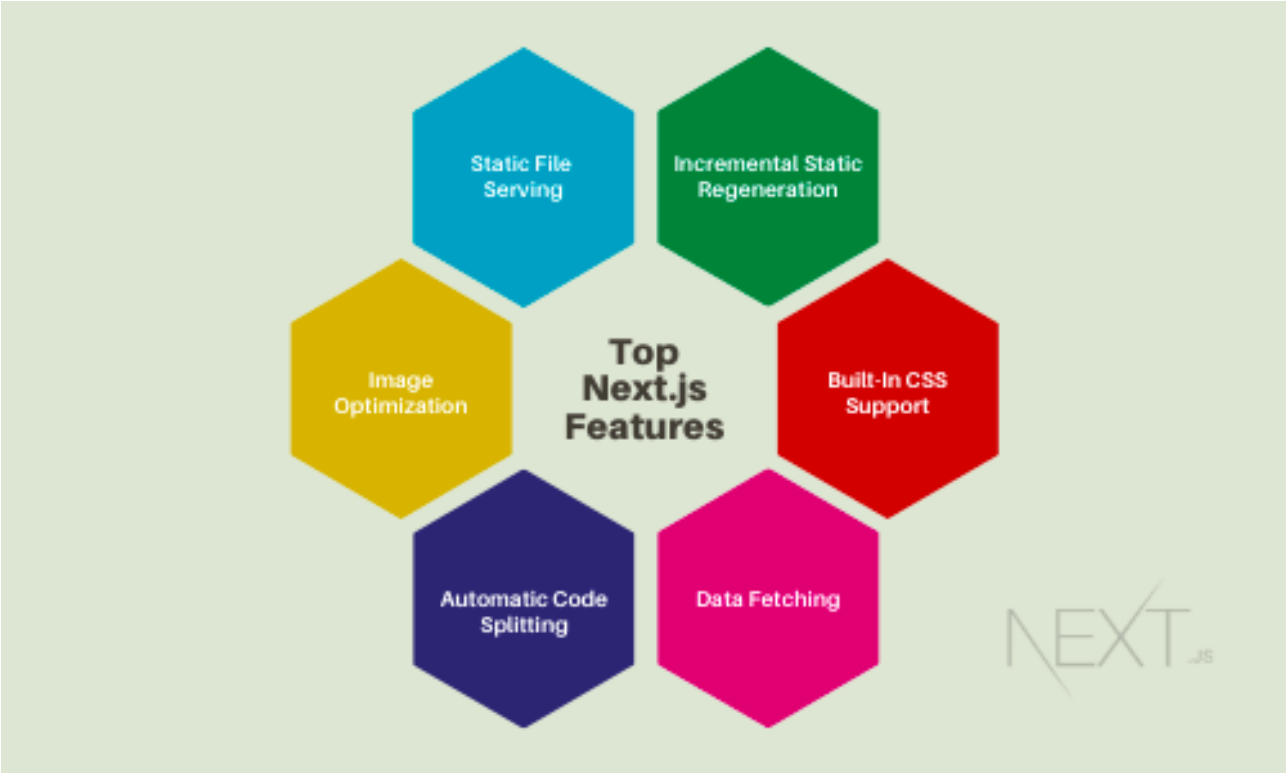
Едно от ключовите предимства в използването на NEXT.js е, че той позволява създаването на високооптимизирани уеб приложения, които зареждат бързо и осигуряват успешен UX. Чрез използване на сървърно рендериране и автоматично разделяне на кода, NEXT.js гарантира, че уеб страниците ще зареждат бързо и ще са напълно интерактивни веднага след като се появят на екрана на потребителя.

NEXT.js включва още и:

- поддръжка на CSS модули, което прави по-лесно управлението на стиловете в големи уеб приложения

- вграден сървър за разработка, който поддържа hot reload (произнася се “хот рилоуд”и означава “гореща замяна” или “бързо презареждане”) на модули и други полезни функции за разработчиците
- компонент за оптимизация на снимки, който преоразмерява графиките до нужната широчина и компресира размера им за по-ниска консумация на данни
- генериране на напълно статични HTML файлове за бързо зареждане
- регенерация на горепосочените статични файлове за постигане на по-високо ниво динамичност
- публична папка за сервиране на файлове до всяка част на приложението, като всеки публичен път до файл започва с наклонена черта “/“ и след това се изписва пътя до самия файл и името му, включвайки и разширението
- вграден API за изграждане на HTTP заявки от всякакъв тип
- вградена поддръжка на TypeScript и ESLint (произнася се “И ЕС Линт”, служи за следене на добри практики)
- компонент за SEO оптимизация, предоставящ възможност за инжектиране на всички нужни мета тагове
- “мързеливо” зареждане или така нареченият “lazy loading” (произнася се “лейзи лоудинг”), за отложено зареждане на по-големи графики или по-сложни парчета код, чието изтегляне би отнело повече време, но тяхната ползваемост е пожелателна, например зареждане на диалогов прозорец при кликане на определен бутон
- вградени методи за “извикване” на данни от API, които се разделят на статични, сървърни и клиентски; статичните генерират HTML файл с всички необходими данни вече изписани в него и го изпращат на потребителя; сървърните генерират този файл на момента

в сървърната част от уеб приложението; клиентските извикват необходимите данни, като след това конструират HTML файла на клиентското устройство



Фиг. 7 Схема на по-важните функции на NEXT.js

GraphQL

GraphQL (произнася се "Граф Кю Ел", идва от Графа или QL, което значи Query Language или език за заявки) е език, чрез който могат да се описват заявки към API чрез синтаксис, наподобяващ JSON. Езикът е разработен от Facebook през 2012г. с цел да предостави по-гъвкава, ефективна и оптимизирана алтернатива на REST (произнася се "рест", акроним на REpresentational State Transfer).

Използвайки GraphQL, разработчикът може да опише точно какви данни желае да му бъдат върнати от сървъра. Възможно е да уточни бройка на върнатите записи, както и филтър, който да премахва нежеланите записи. Така се намалява времето, за което се изпращат данните, както и техният размер.

За да могат да се извличат точни данни, обаче, GraphQL изисква дефинирането на специална схема (schema), която да оказва какво точно се съдържа в API-то. Схемата съдържа много двойки ключ-стойност, техните типове, дефиниции на заявки и мутации (мутациите променят данните, докато заявките само ги извличат), параметри към тях и също така филтри.

GraphQL предоставя и така наречения playground (произнася се “плейграунд”, значи площадка за игра), където разработчикът може да вижда схемата и всички дефинирани мутации и заявки. Там той може да изписва своите заявки с помощта на автоматично довършване (autocomplete) на полетата, които желания обект притежава и подчертаване на синтаксиса (syntax highlight), при грешно или непълно изписване на заявката.

Една хубава и нова функционалност на GraphQL е автоматичното генериране на схема. Разпознавайки вида и версията на базата, GraphQL може да разчете нейните полета и релации и да създаде схема, базирана на тях, допълнена с всички филтри, предоставени от езика SQL. Така разработчикът има възможност да започне работа директно, без да се налага допълнително писане на код.

Node.js

Node.js (произнася се “Ноуд Джей Ес”) е среда с отворен код, в която може да се изпълняват JavaScript скриптове без нуждата от browser. Може да бъде инсталирана на всякакъв вид сървър и на множество операционни системи. Създадена е през 2009г. от Ryan Dahl (произнася се “Раян Дал”) и изградена върху V8 Engine (произнася се “Ви ейт енджин”), който изпълнява скриптовете в browser-а Google Chrome. Противно на смисъла на разширението “js” в името си, Node.js е написана на езиците C и C++, които осигуряват възможността на JavaScript да се изпълнява.

Node.js предоставя на разработчиците един неблокиращ цикъл от събития, базиран на I/O модел. Това означава, че средата работи на принципа вход от потребителя - изход от системата, като всяка следваща команда се вкарва в асинхронен цикъл, който изпраща по-дългите и ресурс-консумиращи задачи за обработка във фонов режим, като така не блокира главната нишка.

Node.js поддържа и високо ниво на разширение. Предоставя много модули и библиотеки, които лесно могат да се интегрират в работния процес.

Node.js се инсталира заедно със менижъра на пакети NPM (Node Package Manager, произнася се “Ен Пи Ем”, “Ноуд Пекидж Мениджър”), който позволява изтеглянето и инсталирането на JavaScript пакети, библиотеки и framework-ве, които могат да се импортират във всеки един js файл в обхвата на проекта.

В съвремието Node.js се използва за създаването на сървърни апликации, които да изпълняват бързо голям брой процеси по-бързо и да боравят с обемни пакети от данни. Също така средата се използва и за създаването на клиентски апликации, в които лесно могат да се инсталират библиотеки от NPM. В някои случаи, създаването на клиентски и сървърни

апликации може да се случва и едновременно в една инстанция на средата node, както е при NEXT.js

Strapi.js

Strapi.js (произнася се “Страпи Джей Ес”) е framework с отворен код за съхранение и дистрибуция на съдържание или така наречения CMS (Content Management System, произнася се “Си Ем Ес”, “Контент Мениджмънт Систем”), който позволява на разработчиците да създадат мощен и бърз API. Създаден е от Pierre Burgy (произнася се “Пиер Бурги”) през 2015г., използвайки средата Node.js и други библиотеки за front-end разработка като React и Vue.js (произнася се “Вю Джей Ес”).

Гъвкавият дизайн на Strapi позволява той да бъде разширяван и допълван доста лесно, за да може да изпълнява нуждите на проекта, в който бива интегриран. Използва езика за заявки GraphQL, който автоматично генерира схема, заявки и мутации на база избора на разработчика във front частта на приложението. След създаване и генерация на заявките, те лесно могат да бъдат допълнени с бизнес логиката на разработваното приложение.

Strapi.js предоставя администраторски панел, от където лесно се менажират всички данни, съдържащи се в базата. Този панел предоставя възможност за инсталация на интеграции към самия framework, позволяващи свързването с други доставчици на услуги като AWS (Amazon Web Services, произнася се “Ей Дабълю Ес”, “Амазон Уеб Сървисиз”), Google Cloud (произнася се “Гугъл Клауд”) и други подобни. Собствени интеграции също могат да бъдат добавяни под формата на plugin-и (произнася се “плъгин”). Възможно е да се създават нови колекции от данни (таблици в базата), да се редактират или изтриват вече съществуващи колекции, да се взимат, създават, редактират и изтриват всички налични записи. В по-новите версии на CMS-а могат да се създават и специални, преизползваеми типове и полета, за по-лесно попълване на данни. Те биват под формата на:

- полета за цвят със специален графичен селектор, изобразяващ цветовата гама RGB
- текстов редактор с всички възможни функции за форматиране на текст или така наречения Rich Text Editor (произнася се “Рич Текст Едитор”)
- поле за автоматична валидация на имейл
- динамична зона, където се предлага избор от много полета и типове данни
- компонент, където могат да се дефинират много полета от различни типове, както и дали самият компонент да бъде повтаряем или не, както и много други.

PostgreSQL

PostgreSQL (произнася се “Пост грес Сикуъл”) е СУБД (система за управление на бази данни) с отворен код, създадена през 1986г. като част от POSTGRES проекта на Изследователски университет в Бъркли, Калифорния, САЩ.

Създаден е, за да предостави мощна и гъвкава платформа за менажиране и манипулиране на обемни пакети от структурирани данни. Може да се използва както и в малки уеб приложения, така и в големи корпоративни проекти.

Отличителен белег на PostgreSQL е поддръжката на сложни SQL функции, като рекурсивни заявки, управление на таблици и аналитични функции.

Друг такъв белег е поддръжката на транзакции, които валидират данните и запазват тяхната цялост, дори и при спиране на захранването, интернет връзката или повреда.

PostgreSQL също може да бъде разширяван с plugin-и и добавки, които да обогатят системата с нови типове, методи на индексиране, оптимизатори и други. Софтуерът е известен и с добрата си защита, която поддържа доста методи на автентикация и механизми за криптиране.

Meilisearch

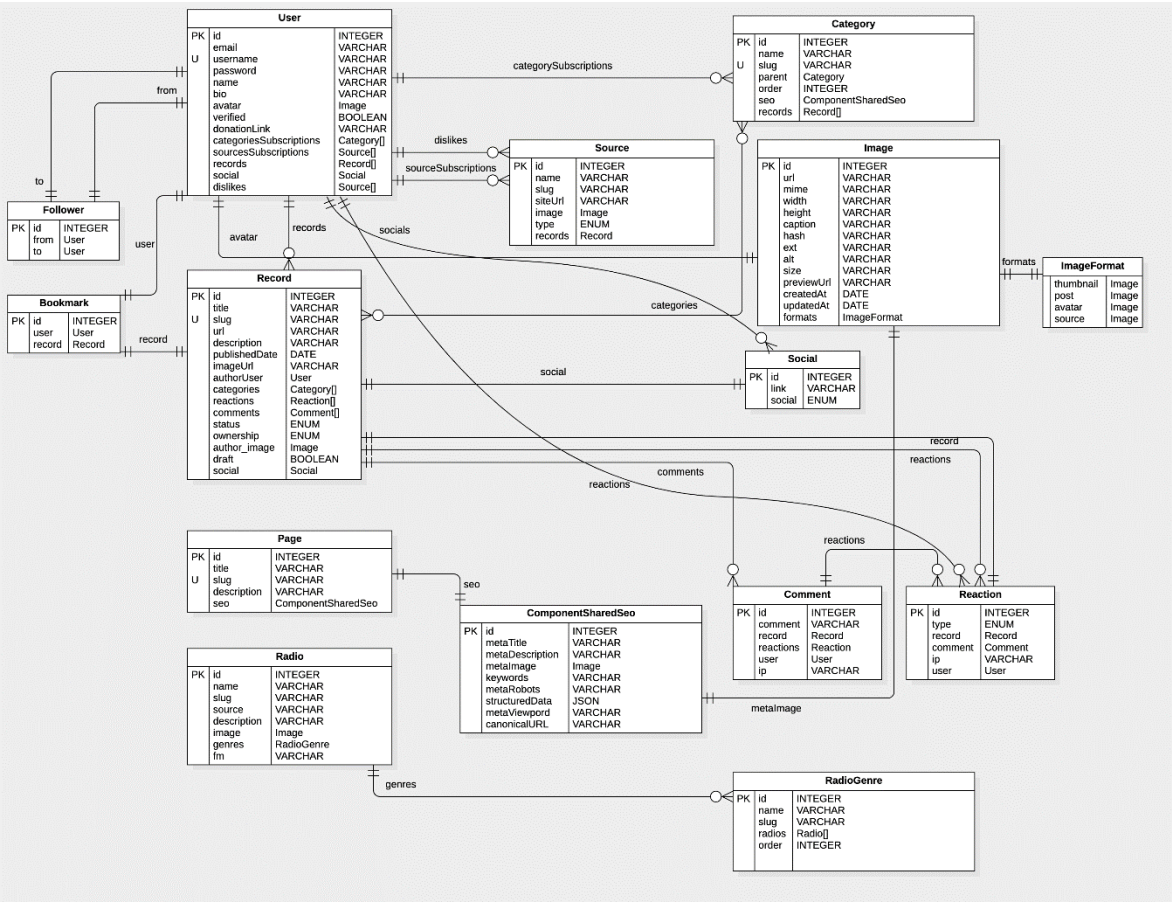
Meilisearch (произнася се “Мейлисърч”) е search-as-a-service търсачка (произнася се “сърч ез ей сървиз”) с отворен код и висока степен на персонализация, която позволява на разработчиците лесно да интегрират търсеща функционалност в приложенията си. Създадена е от Thomas Payet (произнася се “Томас Пайе”) и Clèment Renault (произнася се “Клемент Рено”) през 2018г.

Високата производителност на търсачката позволява да бъдат върнати голям обем резултати за много кратко време. Параметрите, по които се връщат резултатите, могат да бъдат лесно програмирани и филтрирани от разработчика. След всяка промяна по параметрите е нужна реиндексация или презаписване на кешираните от търсачката данни.

Софтуерът може да бъде интегриран лесно във вече съществуващи приложения, написани на различни езици. Заявките към API-а се пишат на синтаксис, който наподобява смеска между REST и GraphQL. Данните, които захранват търсачката могат да бъдат с огромни размери, без това да деградира производителността по никакъв начин.

База данни

Релационна схема на базата данни



Фиг. 8 Схема на базата данни. Таблици и релации между тях

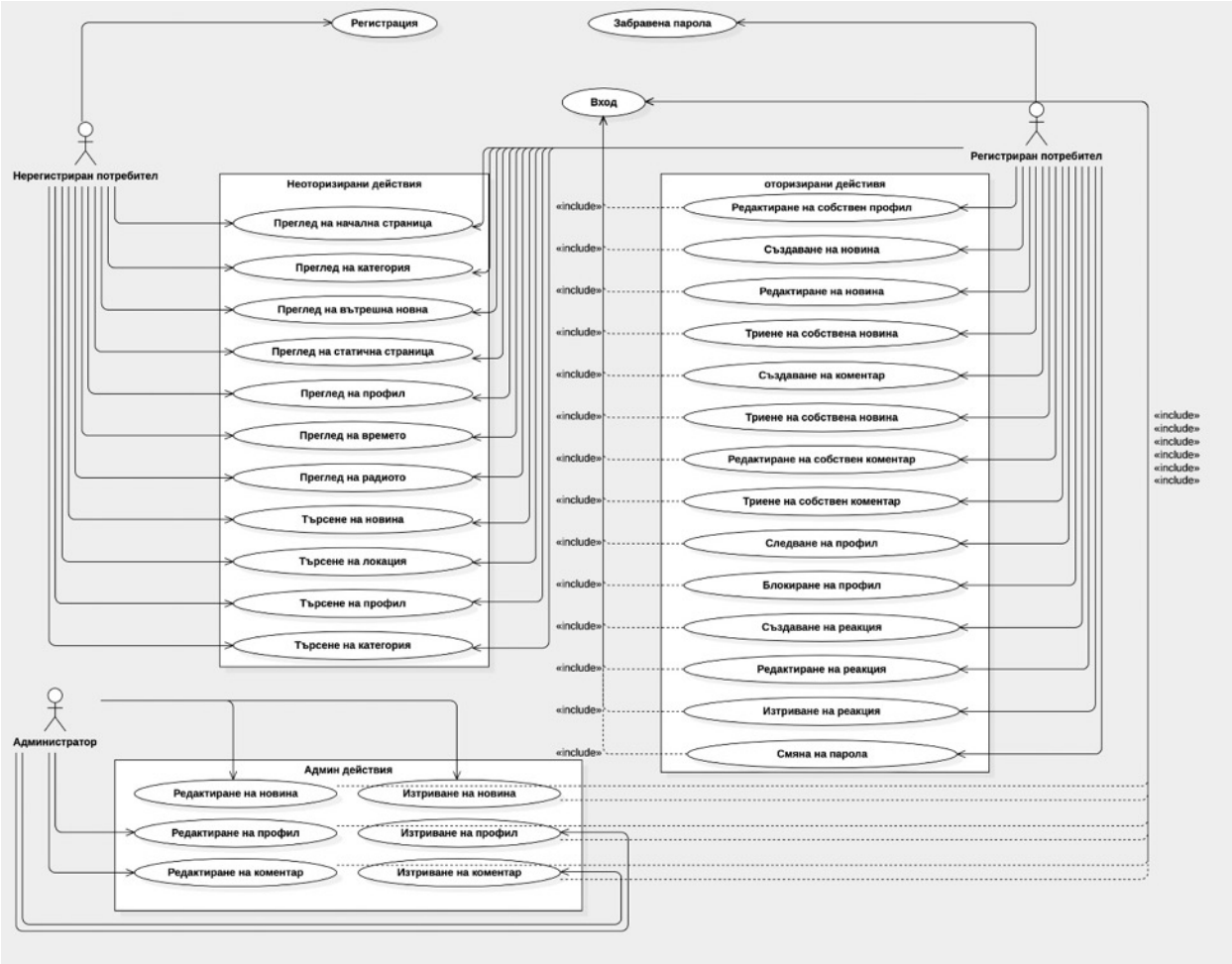
UML Диаграми

Езикът за създаване на диаграми на модели UML (Unified Modeling Language, произнася се “Ю Ем ЕЛ”, “Юнифайд Моделинг Ленгуич”) представлява група от графики, чрез които може визуално да се опише структурата и функцията на един софтуерен продукт.

Създаден е през 1995г. от Grady Booch (произнася се “Грейди Бууч”), Ivar Jacobson (произнася се “Ивар Джейкъбсън”) и James Rumbaugh (произнася се Джеймс Ръмбо). През 1997г. е приет за стандартен език в кръга на отворения консорциум от компании Object Management Group (OMG, произнася се “Оу Ем Джи”, “Обджект Мениджмънт Груп”), които включват гиганти като IBM (произнася се “Ай Би Ем”) и Apple Computer (произнася се “Епъл Къмпютар”). През 2005г. езикът е публикуван от Интернационалната организация за стандартизация (ISO, International Organization for Standardization, произнася се “Ай Ес Оу”, “Интернешънъл Организейшън фор Стандартизейшън”), където се поддържа и до днес.

Случай на употреба (Use case diagram)

Use-case диаграмата изобразява начина на ползване на софтуера от потребителя. В следващата фигура е изобразена use-case диаграмата на приложението.

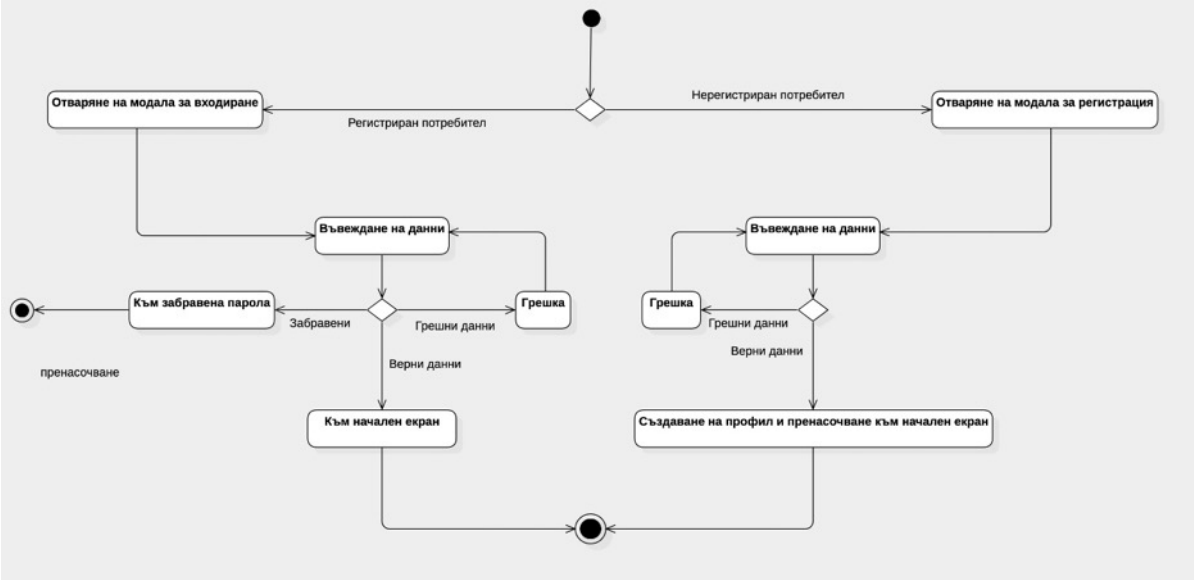


Фиг. 9 Use case диаграма на приложението

Диаграма на дейност (Activity diagram)

Activity диаграмата всъщност е малко по-сложна версия на блок схемата, която изобразява поток от поетапни действия и събития в дадена дейност.

В следващата фигура е изобразена activity диаграмата на процеса на автентикация.

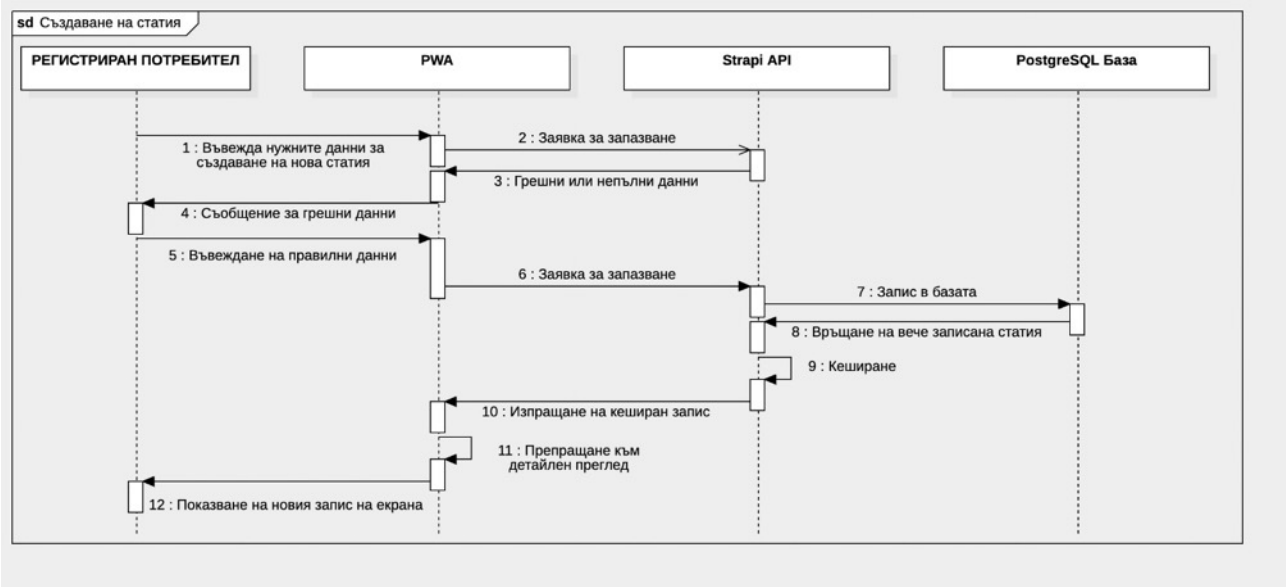


Фиг. 10 Activity диаграма на процеса по автентикация

Диаграма на последователност (Sequence diagram)

Sequence диаграмата показва взаимодействието между слоевете на дадено приложение при изпълнението на дадена дейност. Най-често се изобразява комуникацията на потребителя с front-end частта, която пък от своя страна си комуникира с back-end частта, която може да включва един или много API-и, както и база данни.

В следващата фигура е представена sequence диаграмата на процеса по създаване на статия.



Фиг. 11 Sequence диаграма на процеса по създаване на статия

Адаптивен дизайн (Responsive design)

Адаптивният дизайн или Responsive design (произнася се “респонсив дизайн”) е вид дизайн, при който дадена уеб страница може да адаптира своето оформление и размери на шрифт и графики спрямо големината на устройството, на което се преглежда (мобилен телефон, таблет, настолен компютър и др.). Целта на адаптирането е да се улесни ползването на приложението, взимайки предвид периферията на устройството и начинът, по който се взаимодейства с него.

Във всекидневието си днешният потребител взаимодейства повече със своя мобилен телефон и поради тази причина трябва да се наблегне повече върху мобилната версия на продукта. Тя трябва да предоставя и да изпълнява всяка една функция на настолния вариант, но да съдържа всичко в компактния размер на джобното устройство. За целта е необходимо да бъдат изградени съответните менюта и прозорци, налични в настолната версия, които да бъдат отваряни само когато са нужни. Размерът на интерфейса е задължително да бъде съобразен с физическата големина на телефона. Той не бива да бъде прекалено ситен, понеже взаимодействието с него ще се извършва с пръсти, а не с курсор, но не бива и да бъде прекалено голям, защото ще заема излишно пространство на вече малкия дисплей.

Приложението и администрацията са разработени, спазвайки адаптивния дизайн.

PWA или мултиплатформено приложение

PWA или Progressive Web App (произнася се “Пи Дабълю Ей”, “Прогресив Уеб Ап”) е интернет приложение, изградено чрез вече познатите HTML, CSS и JS, което може да се инсталира на всеки вид устройство, притежаващо уеб браузър и да бъде изпълнявано без да се налага неговото повторно извличане от мрежата.

Едно от предимствата на един софтуер от тип PWA е, че от една код база е възможно да бъде изградено мултиплатформено приложение, което да се изпълнява на iOS, Android, Windows, Linux или друга операционна система само чрез добавянето на няколко реда код. Достатъчно е самата база да бъде качена онлайн и след това да бъде обвита в съответната “опаковка” на операционната система, която да го отваря в уеб браузър без никакви UI елементи (така нареченото web view), а самата функционалност на приложението остава непокътната. След като приложението е вкарано в опаковката си, то може да бъде качено в съответният магазин (App Store, Google Play и др.), където да бъде сваляно от потребителните. В някои случаи то не се различава от нормално native приложение.

Още едно от предимствата е, че при този начин на разработка качеството остава същото, но разходите спадат драстично, понеже е нужно да бъде сформиран само един екип, разбиращ от уеб технологии, докато при конвенционалната разработка са нужни поне три екипа: един за

уеб, един за Android и един за iOS. След изработката и дистрибуцията на пакет за 3-те платформи, софтуерът ще изглежда по абсолютно еднакъв начин навсякъде.

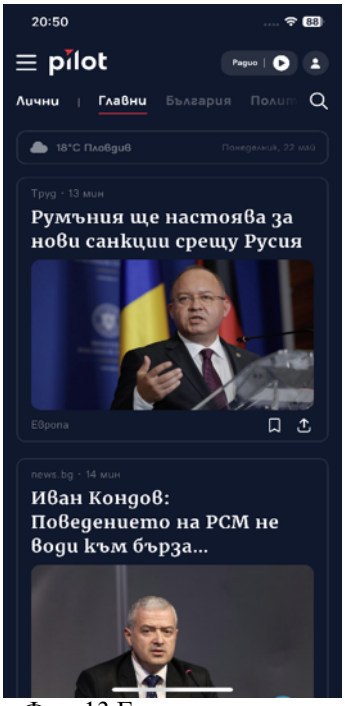
За разработката на приложението е използван NEXT.js, който от своя страна добавя още едно предимство към другите, а именно и поддръжката на offline версия на приложението, която зарежда с последно кеширaните данни. При липса на връзка с интернет пространството, приложението разбира и показва на потребителя последно свалените данни, като, разбира се, го уведомява за липсата на мрежа. Интерфейсът и потребителските настройки, както и сесията на потребителя, са запазени на устройството и продължават да функционират точно като при едно native приложение.

IV. Функционалност

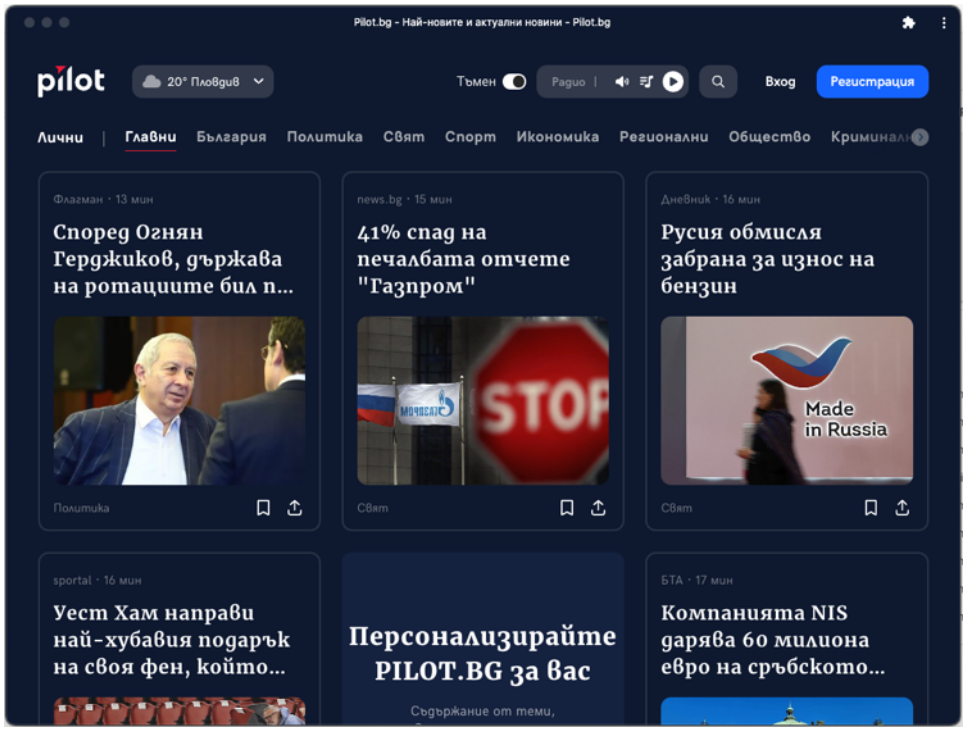
Фиг. 12 Splash Screen

Начален екран (splash screen)

Splash screen (произнася се “Сплаш скрийн”) е екранът, който се появява преди пълното зареждане на приложението. Той съдържа логото на софтуера, за да разбере потребителят, че зарежда точно това, което е отворил. При по-продължително зареждане се появява така наречената “въртялка”, за да индикира някакъв прогрес. Настолното приложение също съдържа splash screen, но уебсайтът - не.



Фиг. 13 Главна страница - гости - мобилно приложение



Фиг. 14 Главна страница - гости - настолно приложение

Главна страница - гости

Върху главната страница за гости на приложението се визуализира feed (произнася се “фийд”) от специално подбрани през администрацията източници, категории и потребители, кеширани през търсачката Meilisearch за по-оптимизирано и бързо зареждане.

В началото на feed-а в мобилното приложение се помещава малка притурка, индикираща текущото време на локацията на която се намираме и датата. В настолната версия тя се намира в header-а.

Header-а за гости съдържа още и: лого (водещо до главната страница), притурка за радио, притурка за входиране и автентикация, търсачка и навигация (съдържаща главните категории на приложението). Настолната версия приютява и превключвателя на тъмна/светла тема.

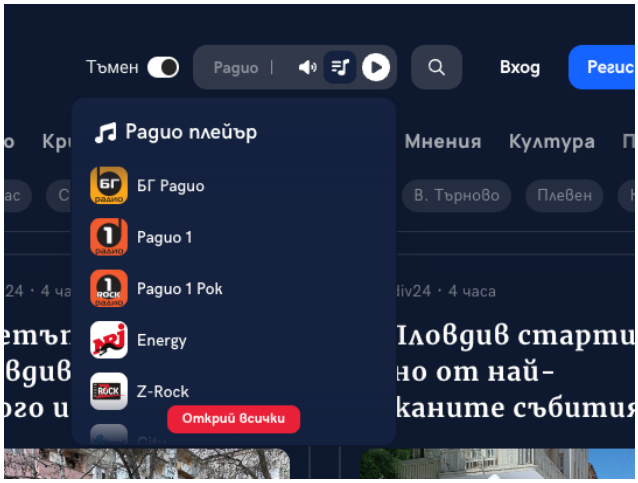
Навигацията представлява списък със странично превъртане (side scrolling), като при кликане на която и да е от имената на категориите в нея, се пренасочва към съответната страница за детайлен преглед на категория.

След натискане на play бутона върху widget-а за радио под мобилно устройство започва да се изпълнява първото в списъка онлайн радио. При клик на целия widget се препраща към страницата радио, където може да се избере друга станция. На настолното приложение списъкът с радиа и влекача за сила на звука изскачат като dropdown менюта.

След натискане на widget-а за прогноза на времето се препраща към страница с детайлен преглед на прогнозата. На настолното приложение има възможност за избиране на друга

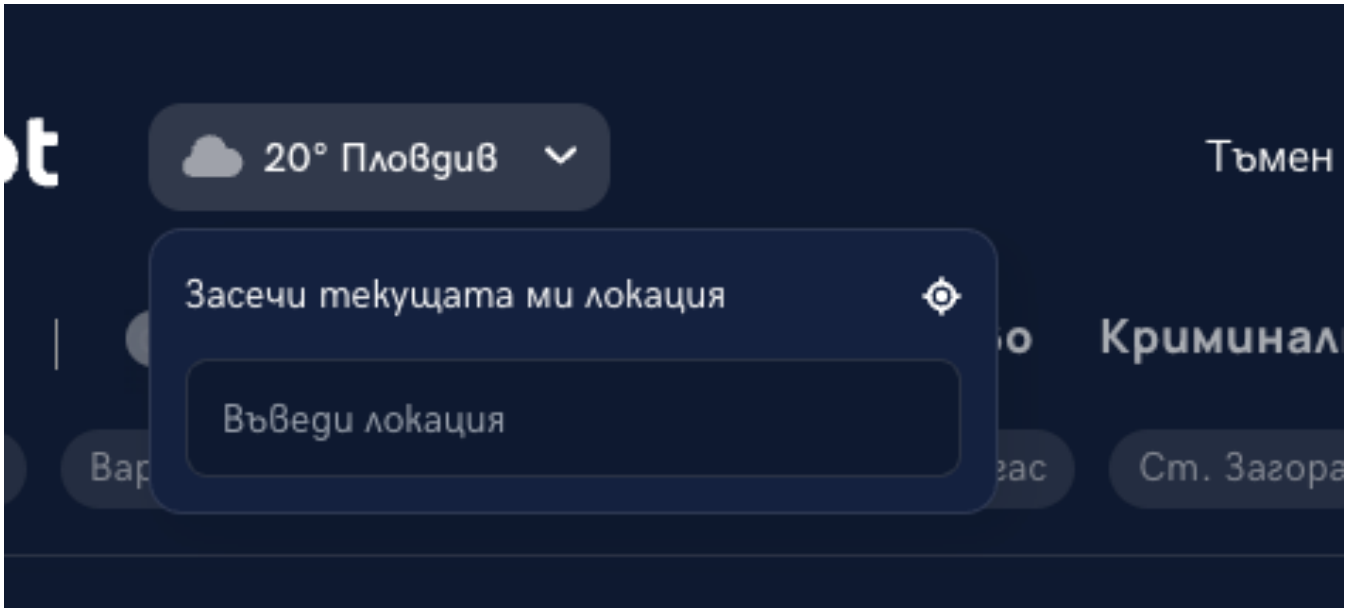
локация още в header частта, докато при мобилно устройство трябва да се достъпи съответната страница за настройка на времето.

При клик на новина от feed-а, тя се отваря в нов прозорец или browser tab в съответния сайт на медията. Ако новината е на потребител, то тя се отваря в специалната страница за детайлен преглед на вътрешна новина.



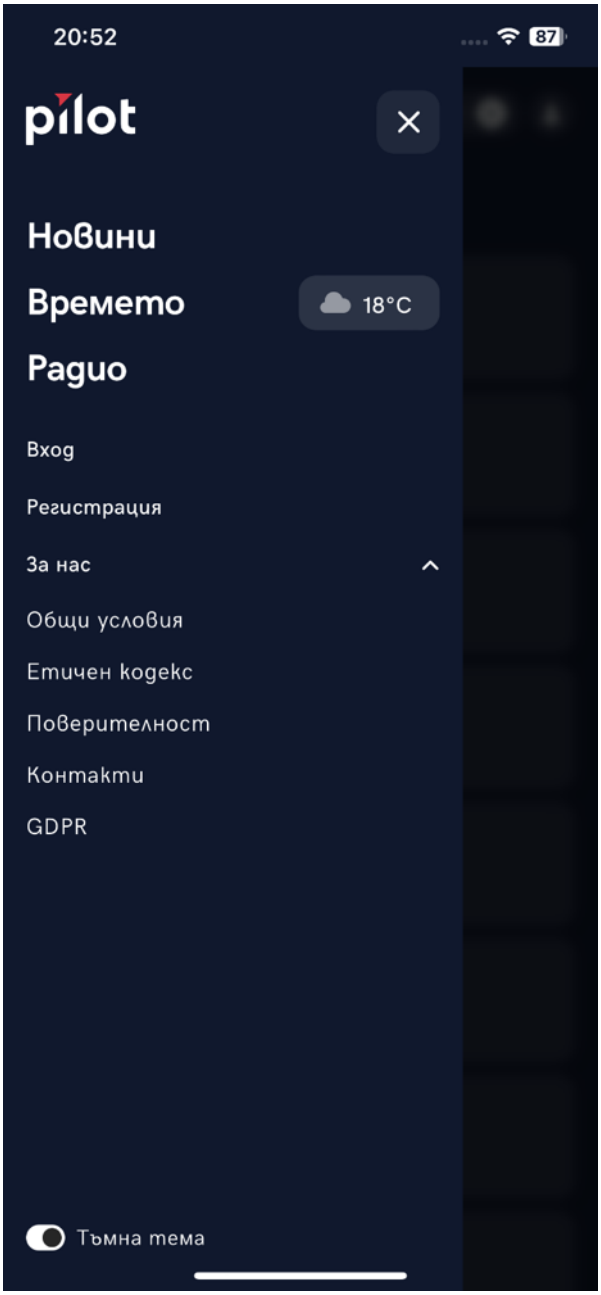
Фиг. 15 Избор на радио под desktop

Фиг. 16 Избор на ниво на звука на радиото под desktop



Фиг. 17 Избор на локация за прогноза на времето под desktop

Екран за навигация под мобилно устройство

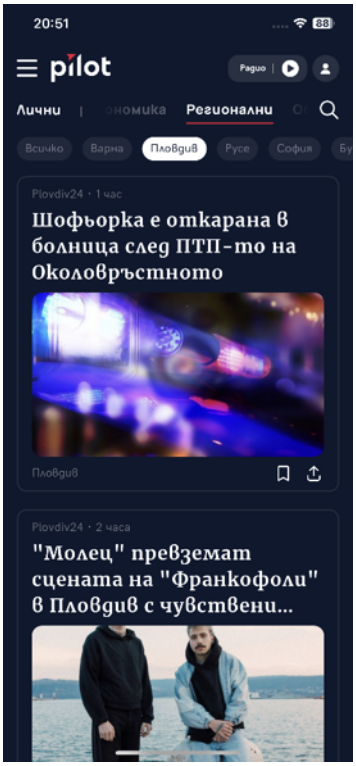


Фиг. 18 Навигационно меню - мобилно устройство

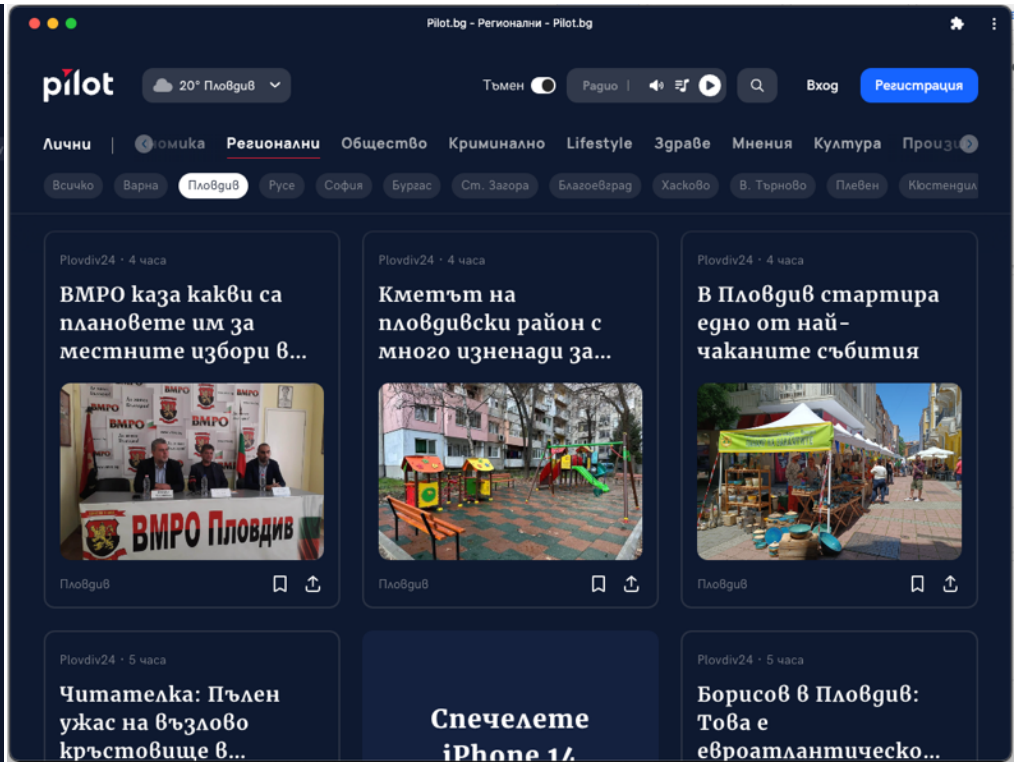
При натискане на бургер меню бутона (трите хоризонтални черти в горния ляв ъгъл) се отваря навигационното меню, от където могат да бъдат достъпени страниците за новини, за прогноза за времето и за радио. В него се съдържат още и бутони за вход и регистрация, както и хиперлинкове към страниците с условия. Най-отдолу се помещава превключвателят на тъмна и светла тема.

Поради размерите си, настолното приложение не притежава такова меню.

Страници на категории и подкатегории - гости



Фиг. 19 Категории и подкатегории - гости - мобилно приложение



Фиг. 20 Категории и подкатегории - гости - настолно приложение

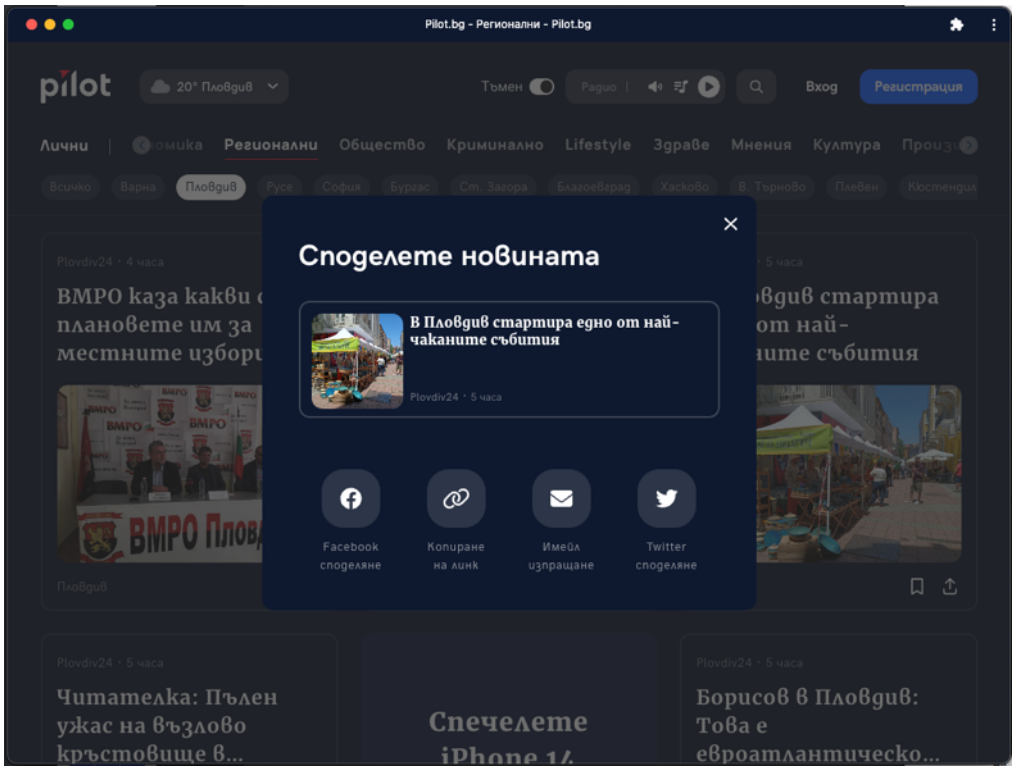
В страниците на категории и подкатегории се помещават само статии, които принадлежат към тях и са от верифицирани медии и/или потребители. Това дали даден потребител или медия може да публикува статии в тези страници се решава от администрацията.

Всяка страница генерира и след това кешира своето съдържание при всяко нейно посещение, за да може при следващо влизане от потребител статиите да се зареждат мигновено. Това кеширане може да се извършва най-много веднъж на всеки 60 секунди. В случай че никой не влезе в страницата за повече от 30 минути, специална функция, създадена да обхожда застоялите страници, ще превалидира съдържанието и ще генерира нов cache.

След определено провлачване надолу по страницата се активира така наречения lazy load на ново съдържание, което се извлича от Meilisearch.

Диалогови прозорци за споделяне на статия

Фиг. 21 Модал за споделяне на статия - мобилно приложение



Фиг. 22 Модал за споделяне на статия - настолно приложение

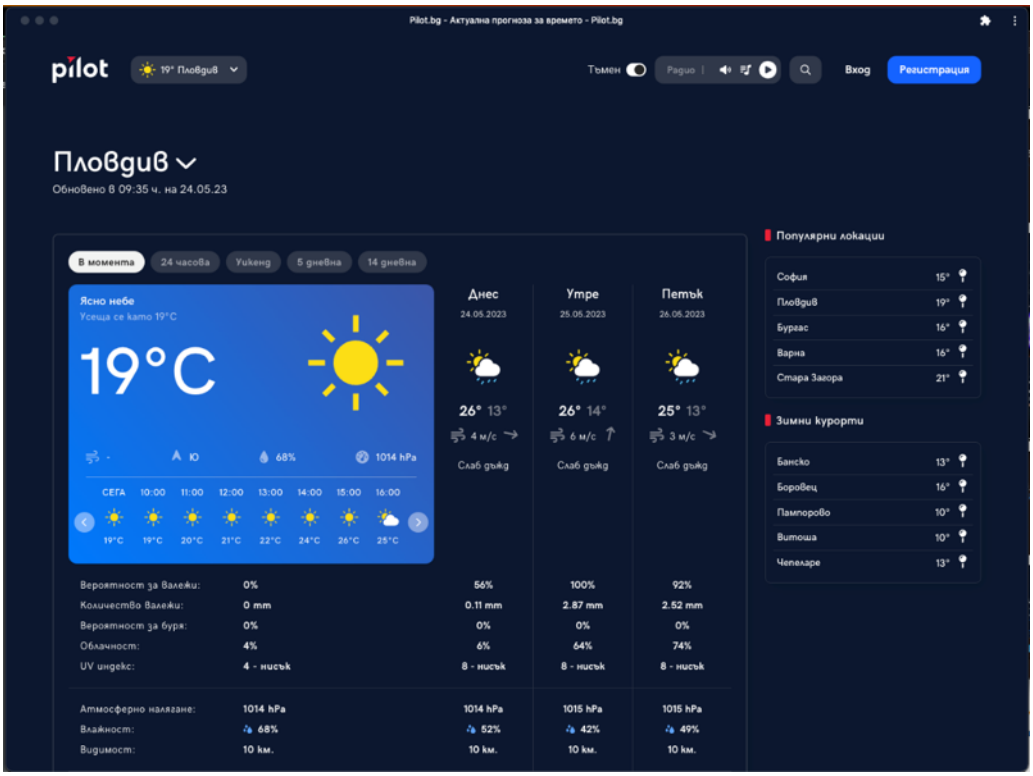
Всеки widget на статия съдържа в себе си бутон за споделяне. Ако потребителят реши да сподели новината, може да натисне този бутон и пред него ще се отвори следният диалогов прозорец.

Появява се малък преглед на новината, която потребителят иска да сподели, а след това ред с 4 бутона:

- Facebook споделяне - при кликане на този бутон се отваря нов прозорец в приложението на Facebook, където автоматично се появява хиперлинк към статията в сайта
- копиране на линк - при кликане на този бутон хиперлинкът се запазва в буферната памет на устройството и може да бъде поставен в друго приложение или текстов редактор
- имейл изпращане - при кликане на този бутон се отваря приложението за изпращане на имейли по подразбиране на устройството, като в него автоматично се попълва хиперлинкът към статията
- Twitter споделяне - при кликане на този бутон се отваря нов прозорец в приложението на Twitter, където автоматично се появява хиперлинк към статията и #pilot

При спонтанно нежелание за споделяне на новината от страна на потребителя, той/тя може да затвори диалоговия прозорец, като кликне извън него, натисне хикса или плъзне надолу с пръст под мобилно устройство.

Фиг. 23 Прогноза - мобилно приложение



Фиг 24. Прогноза - настолно приложение

Страница за детайлен преглед на прогноза за времето

В заглавната част на страницата се намира текущата локация, за която се прогнозира. При натискане на стрелката за надолу (Down Chevron) се отваря падащо меню, подобно на това в header-а, където може да се избере персонализирана локация чрез изписване на името ѝ. След изписване и избиране на локация от списъка с възможни такива се изпраща заявка до Strapi API, който пък от своя страна се допитва до отворения API за прогнози - Open Weather Map (произнася се “оупън уедър мап” означава “Отворена карта за времето”). След извършване на справката се връща последната информация за времето на тази локация, както и прогноза за следващите 14 дни и детайлна прогноза за следващите 24 часа.

В допълнение може да се натисне бутона “Засечи текущата ми локация”, който ще извлече координатите на browser-а и ще ги подаде за обработка на Strapi API. Strapi отново ще се допита до Open Weather Map и след обработка на координатите и допитването до външния

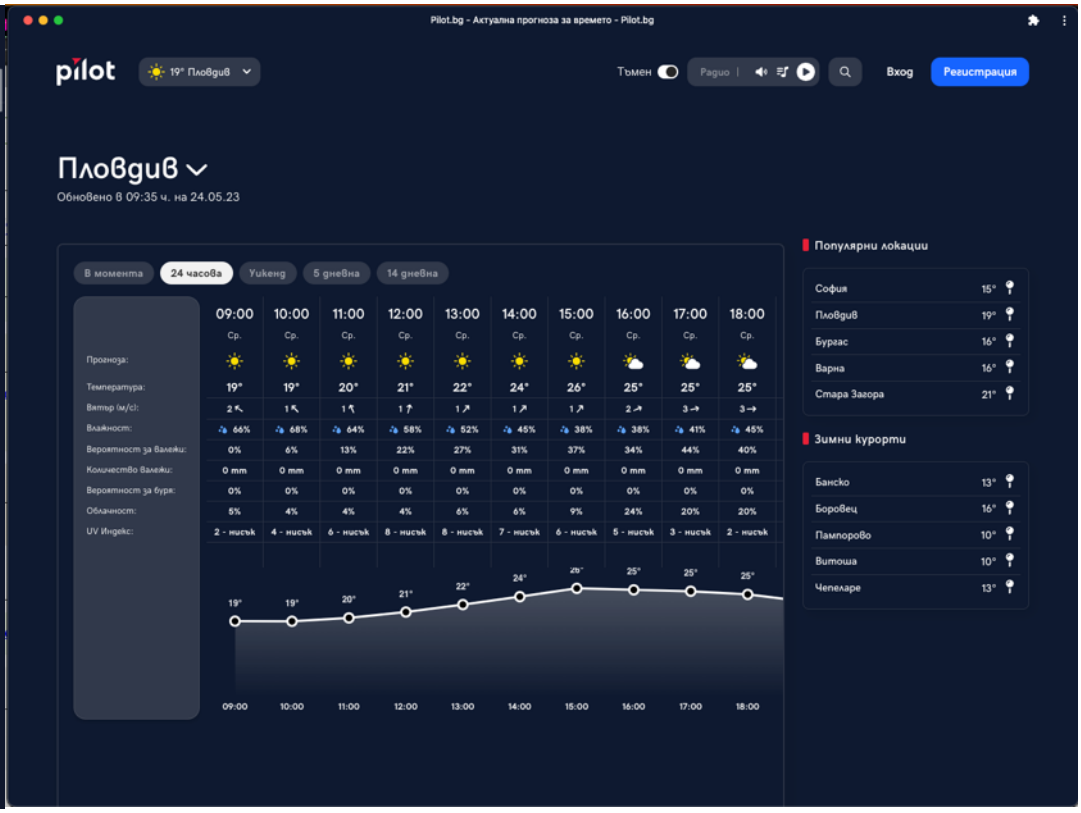
API ще бъде върната прогнозата за най-близката до координатите локация. Първоначалната локация се изчислява автоматично чрез IP адреса на потребителя, който достъпва страницата. Той се подава на отворена база от данни, която връща последно записаната локация на този адрес под формата на географска широчина и дължина. При използване на мобилни данни или динамичен IP адрес са възможни грешки в локацията.

Надолу в страницата е разположен widget-а за текущата прогноза. Той променя цвета, иконата и текста си спрямо времето в момента. Текстът и градусите пристигат чрез API, докато иконите са персонализирани и се показват спрямо уникалния код на времето.

Допълнителна информация за настоящото време, която Open Weather Map връща е скорост и посока на вятъра, влажност и налягане, които се помещават на реда точно под голямата иконка.

Под тях се намира детайлна прогноза за следващите 24 часа, вмъкната в контейнер със странично превъртане, придружена с температура и иконка за атмосферните условия, уникални за всеки час. Там също може да се открие и часът на следващият изгрев/залез.

Фиг.25 Подробен преглед на ден - мобилно приложение



Фиг. 26 24 часова прогноза - настолно приложение

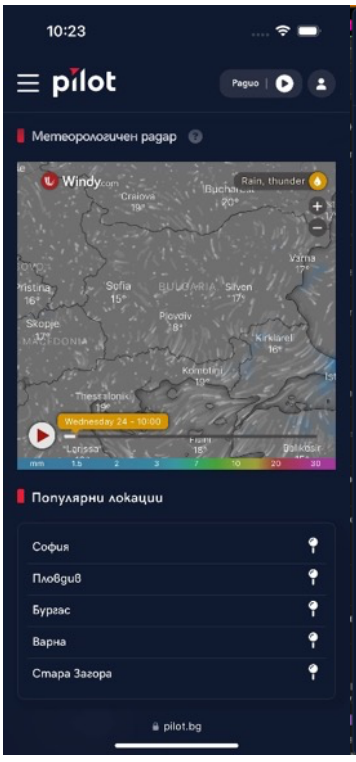
Около този widget в двете версии на приложението може да се открие още допълнителна информация за фактори като:

- вероятност за валежи

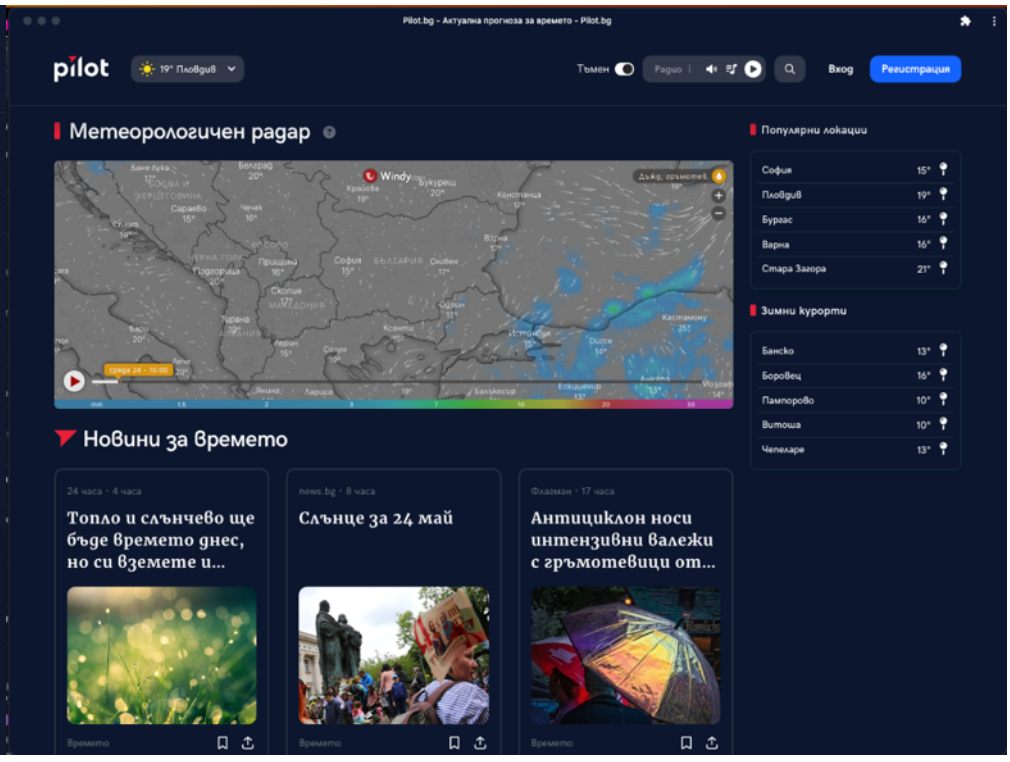
- количество валежи
- вероятност за буря
- процент облачност
- UV индекс
- видимост
- време до залез
- продължителност на деня и други

Под мобилно устройство тази информация може да бъде достъпена при клик на “Днес”, “Утре” или някой друг ден от списъка, докато при настолното приложение тази информация е достъпна веднага без никакви манипулации по оформлението.

Настолната версия разширява функционалността за 24-часова прогноза, като изнася данните в нов tab, където по-обстойно се описват подробностите чрез гореописаните фактори. В допълнение, настолната версия предоставя и още 3 tab-а: прогноза за уикенда, прогноза за 5 и 14 дни напред. Дневните прогнози притежават и един допълнителен фактор - фаза на луната.



Фиг 27. Други компоненти на прогнозата - мобилно приложение



Фиг. 28 Други компоненти на прогнозата - настолно приложение

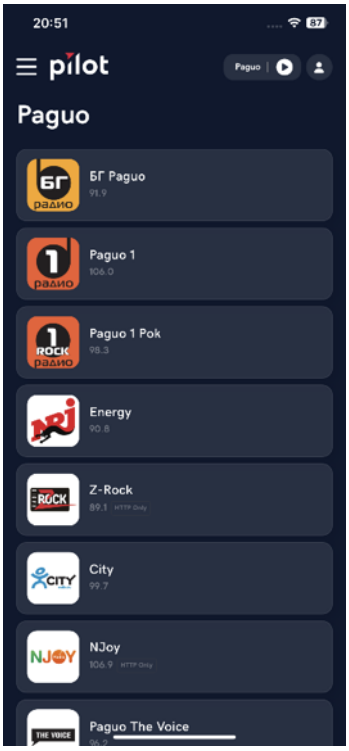
В зависимост от устройството, разпределението на следващите 3 компонента се различава, но редът остава същия, а именно:

• Списък с популярни локации - sticky списък на настолната версия (стои със съдържанието през цялото време), но вграден с главното разпределение в мобилното приложение. При клик на някоя от локациите се извършва проверка, подобна на тази за търсене на персонализирана локация и данните се обновяват с тези за новоизбраната локация. При желание за смяна, нова локация може да се потърси от header-а.

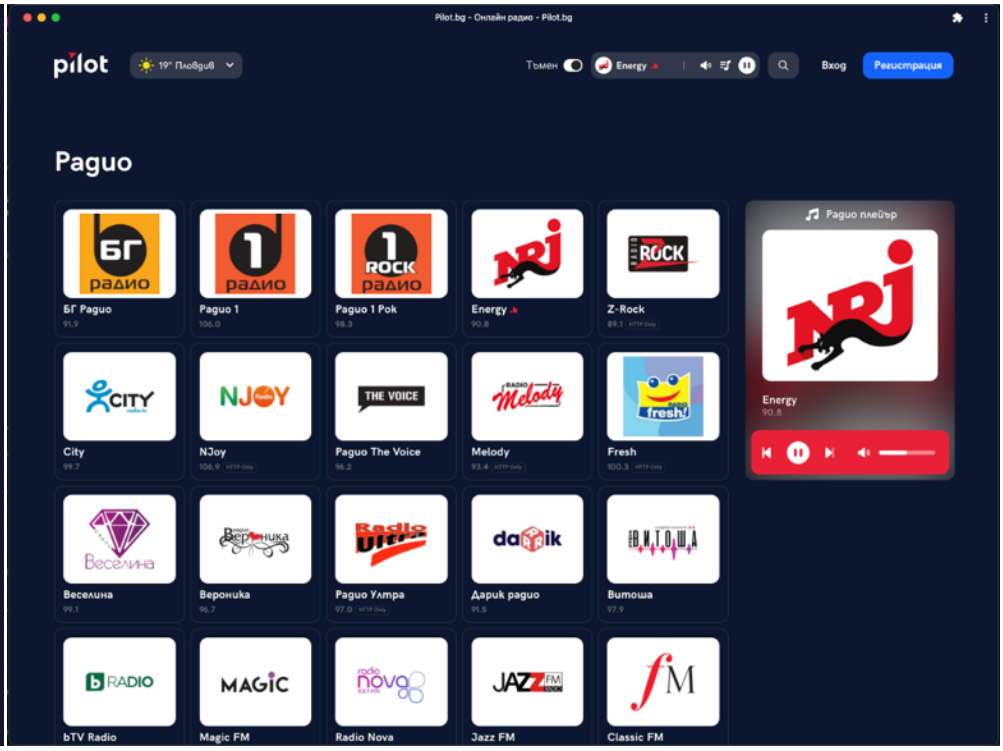
• Вграден widget в рамка на windy.com - widget-ът е вграден чрез iFrame HTML таг и показва визуално и върху карта въздушните течения в района. При желание от потребителя, визуализацията може да се смени с такава, която показва: дъжд, натрупване, бури и други атмосферни условия.

• Списък със статии от категория “Времето” - статиите са същите, като от страницата за детайлен преглед на категория “Времето”, но за удобство на потребителя са изкарани и тук.

Страница за детайлен преглед на радио



Фиг. 29 Страница радио - мобилно приложение



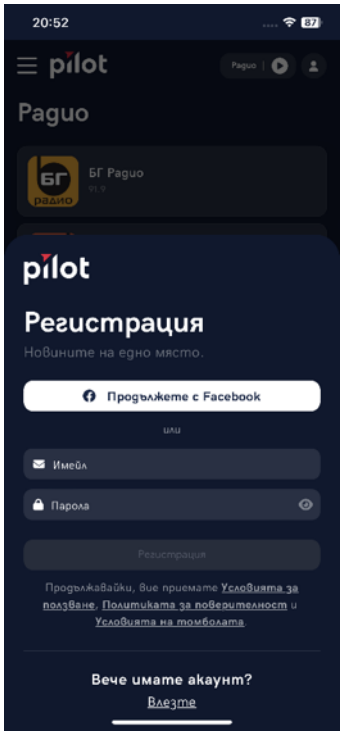
Фиг. 30 Страница радио - настолно приложение

Страницата за детайлен преглед на радио съдържа списък с онлайн радиа. Всеки обект от списъка е бутон, съдържащ името, честотата на предаване и логото на съответното радио.

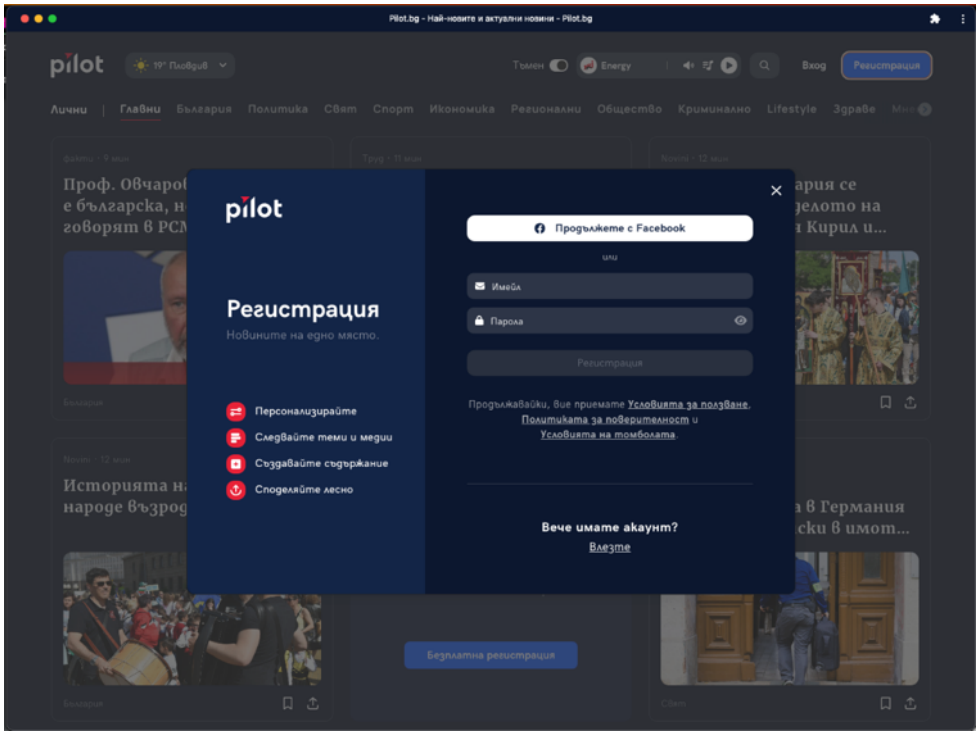
При кликане на радиото се изпраща заявка за поток от данни, чрез който може да се възпроизведе звук. Всяко радио има уникален онлайн адрес, от който се черпят данните. За възпроизвеждане и контрол над потока е използвана библиотеката React Player, която предоставя възможност за пускане, спиране, сменяне на поток, увеличаване и намаляне на звука, заглушаване и индикация за изпълнение/пауза.

След натискане се появява индикатор за зареждане (въртялка) върху самия бутон, в widget-а в header-а и върху самия player, която се заменя с индикатор за изпълнение след успешно свързване с радиото. Индикаторът за изпълнение е 3 червени вертикални черти, променящи височината си в ритъм.

Диалогов прозорец за регистрация



Фиг. 31 Регистрация - мобилно приложение



Фиг 32. Регистрация - настолно приложение

При кликане на който и да е бутон в разпределението, индикиращ регистрация чрез текст или иконка, или при опит за извършване на действие за регистрирани потребители, се отваря диалогов прозорец, подканващ потребителя да си създаде потребителски профил.

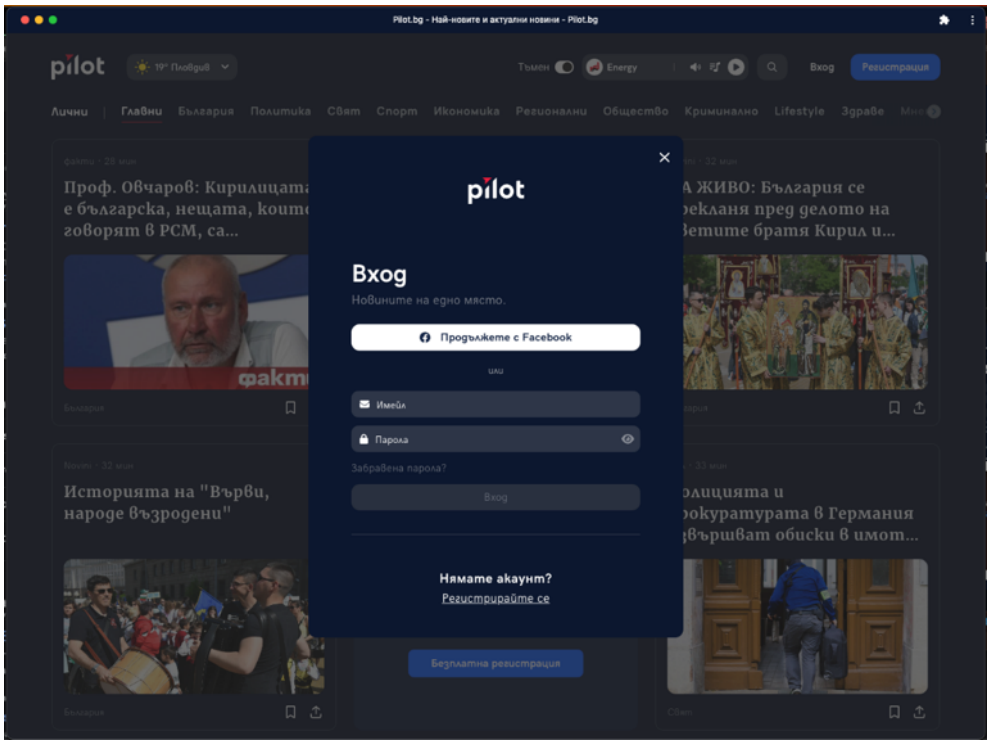
Потребителски профил може да се създаде по два начина: първият е входиране през Facebook, а втория - чрез ръчно въвеждане на данни.

За регистрация с Facebook е нужно първо да се създаде приложение във Facebook developer console и да се свърже с текущото уеб приложение, като се разреши неговият домейн и се добави адрес за пренасочване, където ще се осъществи автентикацията между Facebook и Strapi API. След нужните разрешения се натиска бутона за регистрация в уеб приложението. Потребителят бива пренасочен към уеб приложението на Facebook, където потвърждава профила си. След успешно потвърждение Facebook пренасочва обратно към уеб приложението със зареден потребителски token. Уеб приложението изпраща този token към Strapi API, където се извършва автентикацията. Ако потребителят има създаден профил, свързан с този acebook акаунт, се връща JWT token-ът на акаунта. Ако потебителят няма акаунт през този provider, се създава нов акаунт, като се извлича имейла, името и профилната снимка от Facebook и се връща JWT token на уеб приложението. След получаване на token, софтуерът го записва в бисквиткие, с което процесът по автентикация приключва и клиентът може да ползва функционалностите като регистриран потребител.

За ръчна регистрация процесът е подобен. Потребителят изписва валидни имейл и парола. Ако са невалидни ще излезе грешка, под формата на червен надпис под съответното поле. След успешно въведени данни се изпраща заявка за създаване на профил с имейл и парола. Данните се записват, като паролата задължително се криптира. При успешен запис се връща JWT token на уеб приложението. При грешка, съответното съобщение се връща към потребителя под формата на модал с подкана да опита по-късно, а грешката се записва в база данни за грешки, наречена Sentry.io.

Диалогов прозорец за входиране

Фиг. 33 Екран за входиране - мобилно

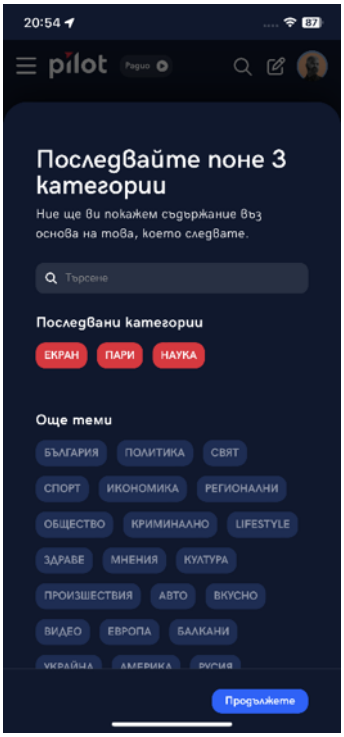


Фиг 34. Екран за входиране - настолно приложение

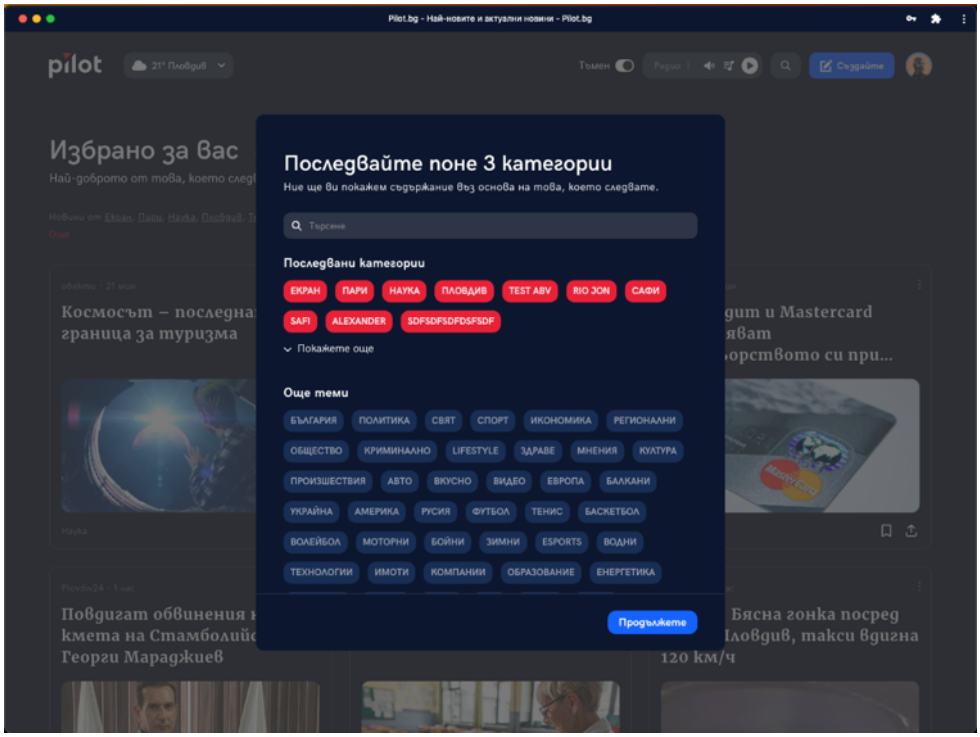
При натискане на бутона “Влезте” в модала за регистрация или при кликане на който и да е бутон “Вход” се зарежда диалоговият прозорец за входиране. Процесът по въвеждане на данни е идентичен с този на регистрацията. Може да се кликне “Продължете с facebook” и ще се извърши гореописаният процес, но може и ръчно да се въведат данни на вече съществуващ профил. При въвеждане на грешни данни или данни на несъществуващ профил, ще се върне грешка, че имейлът и/или паролата на потребителя са грешни.

При въвеждане на коректни данни се прави заявка за вход и се връща JWT token, който се записва в бисквитките.

Екран за следване на категории



Фиг 35. Екран за следване на категории - регистриран потребител - мобилно приложение



Фиг 36. Екран за следване на категории - регистриран потребител - настолно приложение

След регистрация на потребител (или при натискане на бутон “следвани” от екран “Потребителски feed”) се отваря диалогов прозорец, който подканва потребителя да избере поне 3 категории от списъка с теми. Изборът трябва да съдържа поне 3, за да може да се изгради адекватен feed от статии, които потребителят може да разглежда.

След клик на неактивен елемент от списъка, той се оцветява в червено, индикирайки, че при следващо запазване, той ще бъде отбелязан като последван.

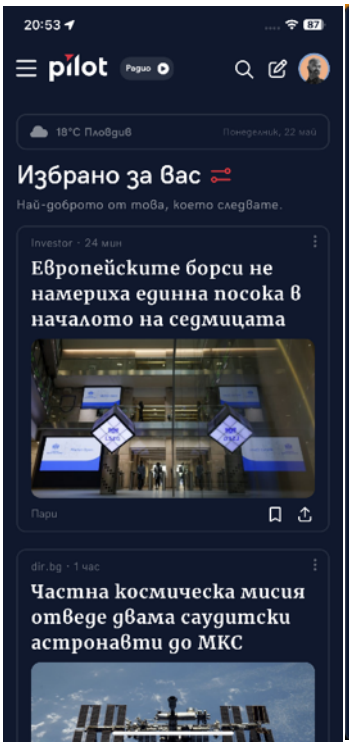
Обратно, при клик на активен елемент, той се обезцветява, обозначавайки се като отследван.

При натискане на бутон “Продължете” се изпраща заявка към Strapi API, където се запазват новите данни на потребителя. След това се извършва презареждане на страницата, за да могат да се наберат нови статии за feed-а.

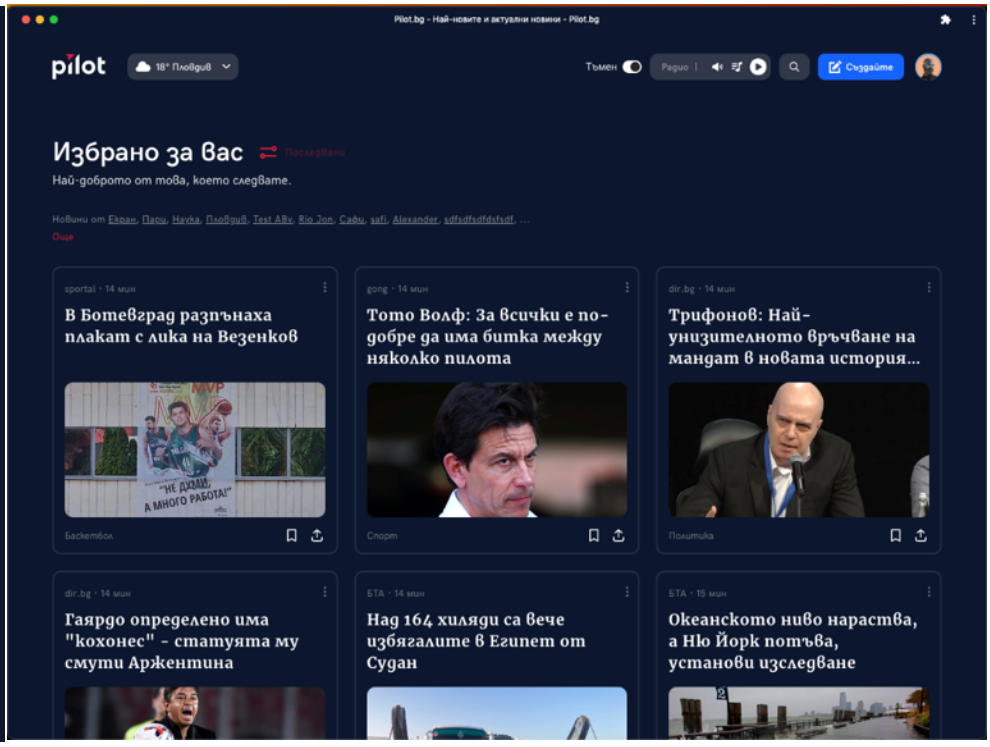
При изписване на символи в търсачката, непоследваните теми се филтрират.

При натискане на бутон “покажете още” се отваря целият списък с последвани теми. В него фигурират и последваните потребители, както и последваните медии. Това е така, за да може потребителят по-бързо да нанесе нужните корекции в интересите си, ако дадена тема вече не му допада.

Екран потребителски feed



Фиг 37. Потребителски feed - мобилно приложение



Фиг 38. Потребителски feed - настолно приложение

След регистрация и избиране на поне 3 теми за следване или след входиране се зарежда екранът “Потребителски feed”. В този екран, благодарение на търсачката Meilisearch и нейната способност да кешира огромно количество данни и да ги филтрира за секунди, се зареждат новини само от следваните източници, които потребителят е задал в интересите си.

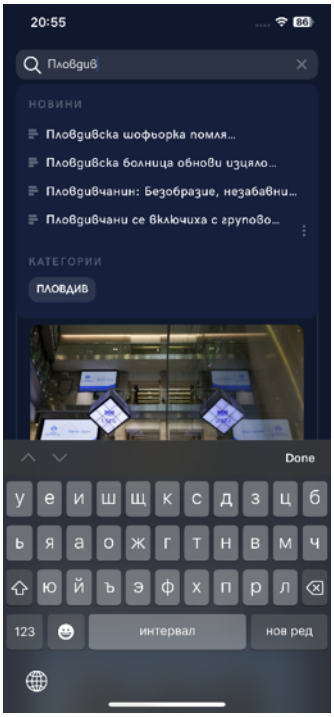
В първата част на екрана се помещава заглавие “Избрано за вас”, което индикира, че това е персонален feed. До него се намира бутонът за редакция на интересите, който отваря диалоговия прозорец за следване на категории.

Под тях фигурира списък с вече последваните източници в по-кратък и ситен вариант, като всеки елемент от списъка е хиперлинк към съответната категория, подкатегория, профил на медия или потребителски профил. При натискане на бутона “още” се показва пълният списък с източници, понеже той може да бъде доста обемен.

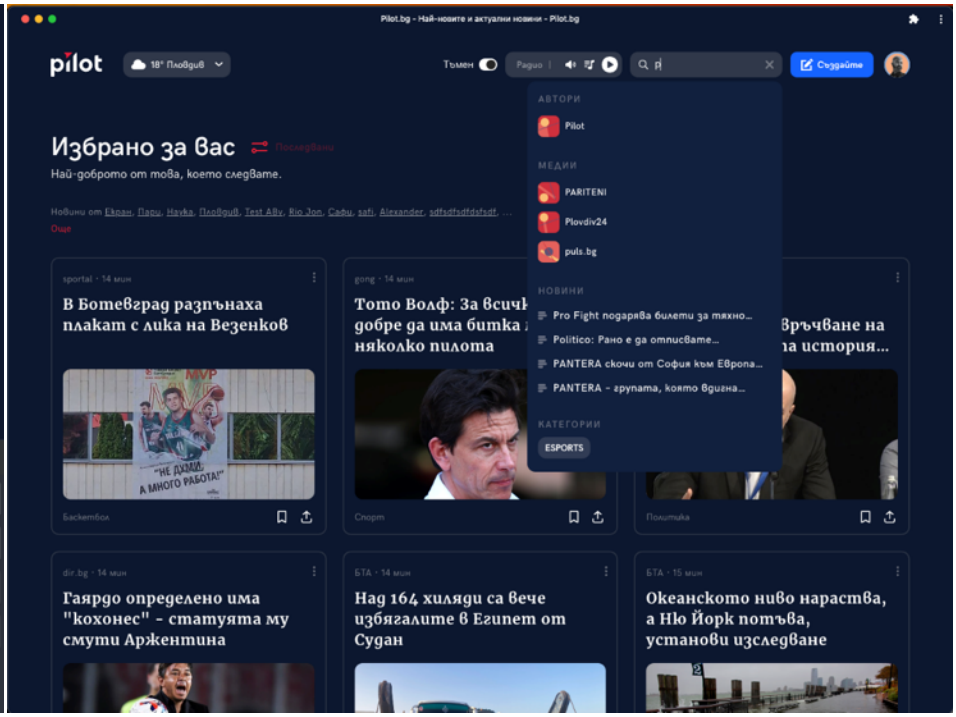
Набива се на око, че header-ът вече е променен и липсва сложната навигация с категории и подкатегории. Това също е белег, показващ, че потребителят вече има профил. При желание да види нови категории, различни от следваните или пък да посети профил на медия или друг потребител, той може да използва търсачката на Meilisearch, като натисне иконката с лупичка.

При scroll надолу по feed-а се зареждат нови статии, чрез lazy load и Meilisearch.

Dropdown за търсене



Фиг. 39 Dropdown за търсене - мобилно приложение



Фиг. 40 Dropdown за търсене - настолно приложение

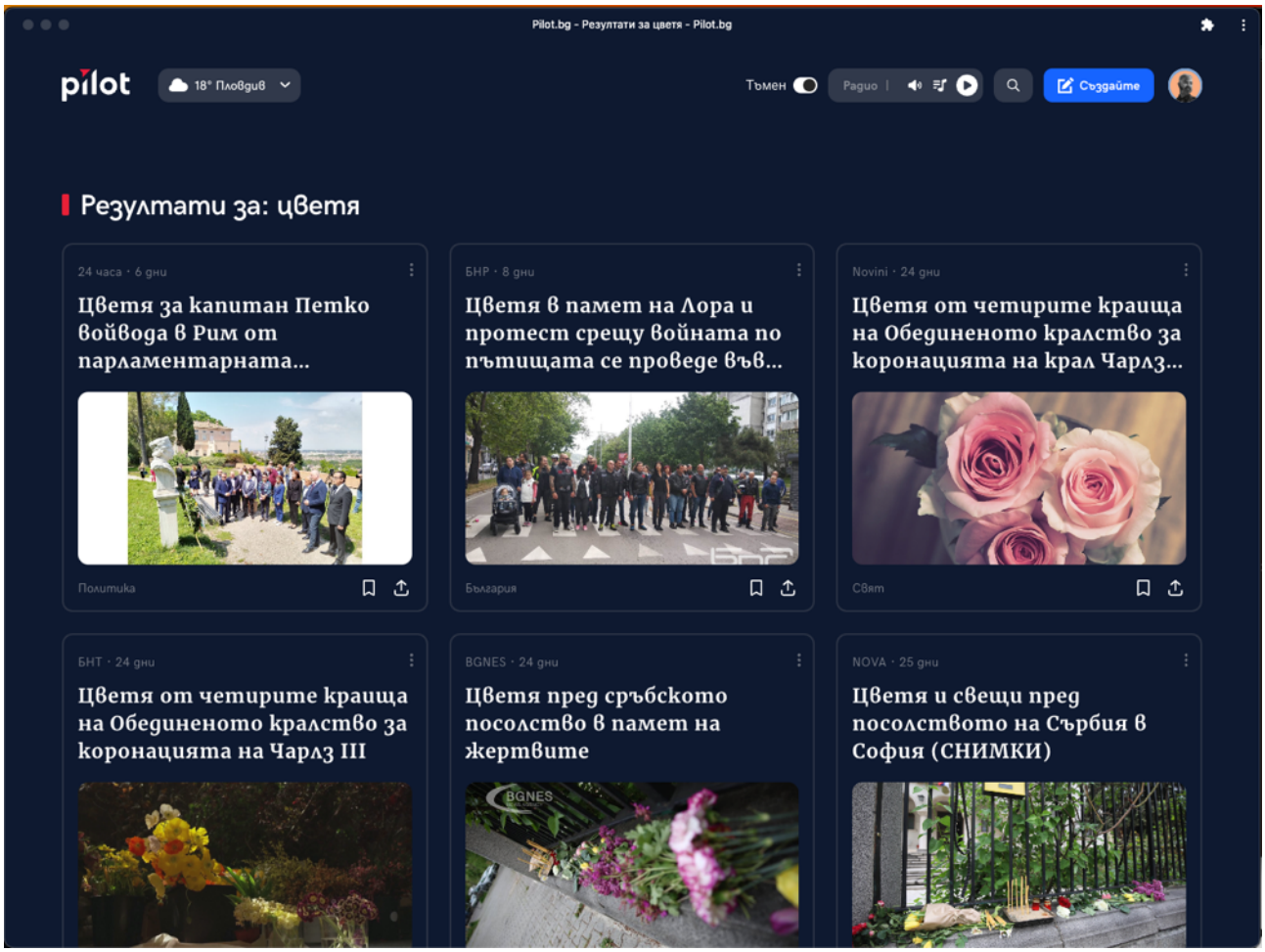
Dropdown-ът за търсене се появява при клик на иконка, в която има лупа. При натискане карето с лупата се разширява, разкривайки вече фокусирано поле за писане. При мобилната версия излиза и екранната клавиатура.

При изписване на символ в полето се появяват индексирани от Meilisearch резултати в 4 различни категории: автори или потребители, медии, новини и категории. Ако няма нито един резултат за някоя от секциите, цялата секция се скрива. Обикновено това са всички без тази за новини. При изчерпване на резултатите и от секция новини се показва надпис “Няма намерени резултати за …”.

При натискане на бутона Return (ентер) приложението навигира до страница с всички индексирани новини, съдържащи израза, написан в полето за търсене. В тази страница не могат да бъдат открити резултати за категории, медии или автори, но могат да бъдат достъпени, чрез хиперлинковете в новинарските карти.

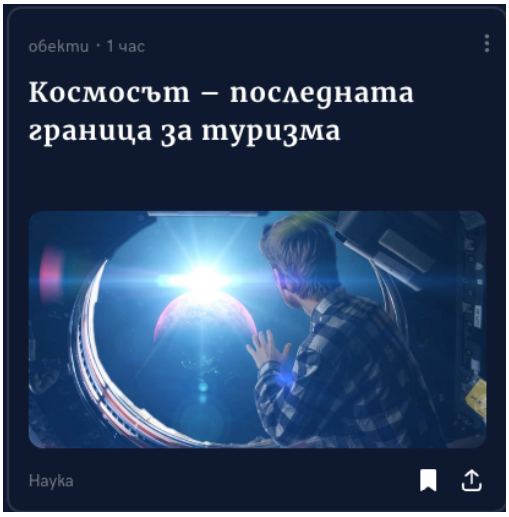
При клик на който и да е елемент от списъка с резултати се извършва препращане към съответният ресурс в уеб приложението. Ако избраният елемент е външна новина, то се отваря нов tab, в който се зарежда страницата на новината в портала на нейната медия.

След избор на резултат, полето за търсене се зачиства и прибира обратно зад иконката на лупа.



Фиг. 41 Екран на резултати от търсене

Индикатор за запазена новина

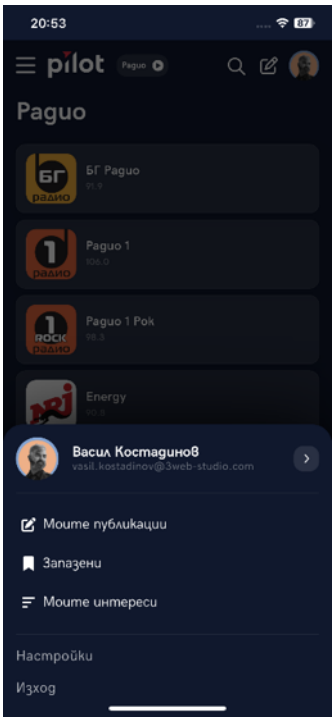


Фиг. 42 Индикатор за запазена новина

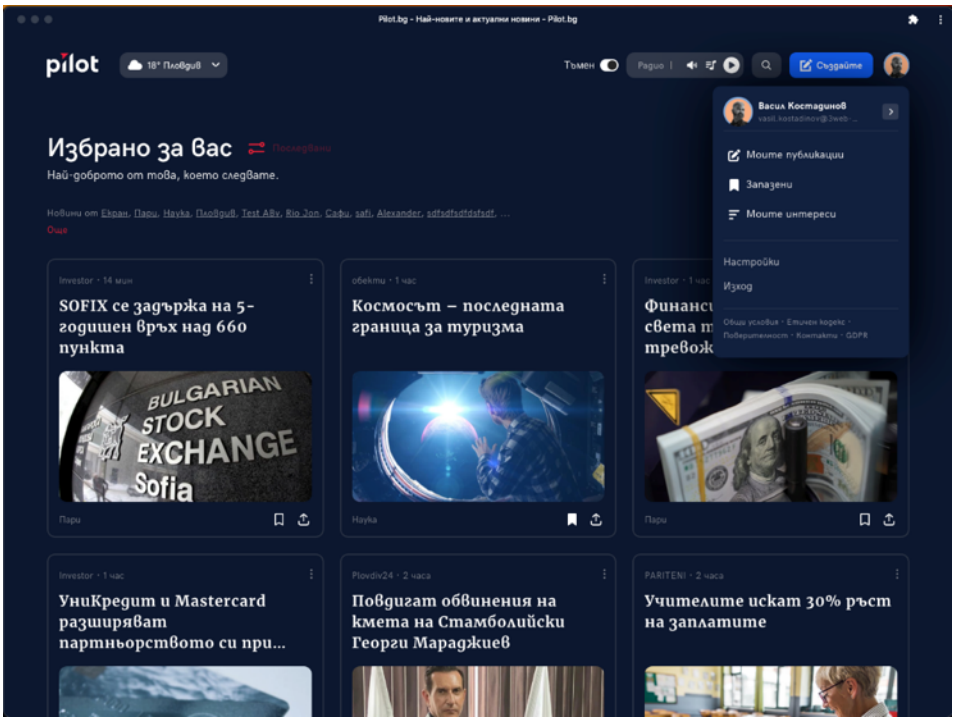
След натискане на отметката за запазване (bookmark, произнася се “буукмарк”), при регистриран потребител се изпраща заявка до Strapi API, където в отделна таблица се запазват id на новина и id на потребител. След успешно изпълнение на заявката се връща id на новия bookmark запис и на клиентската част отметката се запълва в бяло, давайки знак, че новината е запазена. Запазените новини могат да бъдат открити в едноименната секция в личния потребителски профил.

При натискане на вече запълнена иконка се изпраща DELETE заявка с id на bookmark-а и JWT token на потребителя. Той може да изтрива само bookmark-ве, съдържащи неговото id. Проверката се извършва с гореспоменатият JWT token.

Popover за достъп до личен потребителски профил и настройки



Фиг. 43 Popover с лични данни - мобилно приложение



Фиг 44. Popover с лични данни - настолно приложение.

Този popover (буквален превод “появява се отгоре на нещо”) се появява при клик на аватара в header-а и дава на потребителя възможност да достъпи до своя профил в платформата. На първо място стои неговият аватар, редом с името му и имейлът, използван за входирането. Най-вдясно фигурира стрелка, индикираща, че цялата секция е хиперлинк, който ще изпрати потребителя до профила му.

Под заглавната част се помещават 3 линка, които извършват същата работа като нея, но изпращат потребителя до една от 3-те секции на неговия личен профил, а именно: публикациите или статиите, които е написал и/или запазил като чернови, запазените му новини, записани чрез натискана на bookmark отметката и неговите интереси - страница с подробни данни за следваните източници.

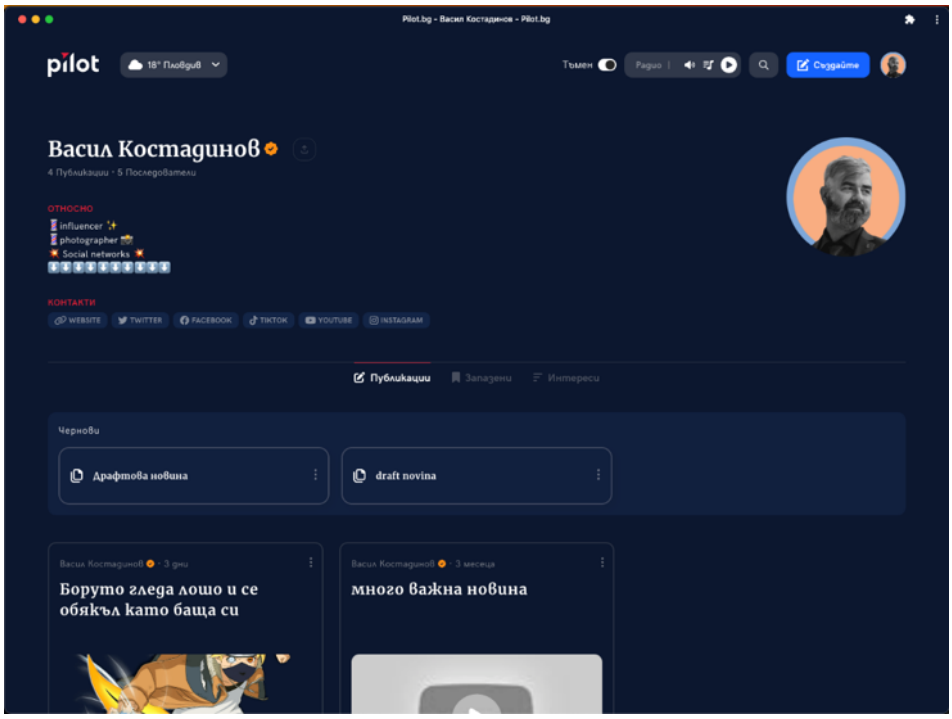
Под тези елементи се намират бутона за настройки, отварящ екран за редакция на личните данни и бутона за изход, който изтрива JWT token-а от бисквитките, потребителският кеш, запазен на устройството и пренасочващ потребителя към началната страница.

Най-отдолу се намира списък с всички легални страници: Общи условия, Етичен кодекс, Поверителност, Контакти и GDPR.

Под мобилно устройство, popover-ът се отваря само до бутона за изходиране. Легалните страници се появяват при допълнителен скрол.

Личен потребителски профил - секция “Моите публикации”

Фиг. 45 Моите публикации - мобилно



Фиг. 46 Моите публикации - настолно приложение

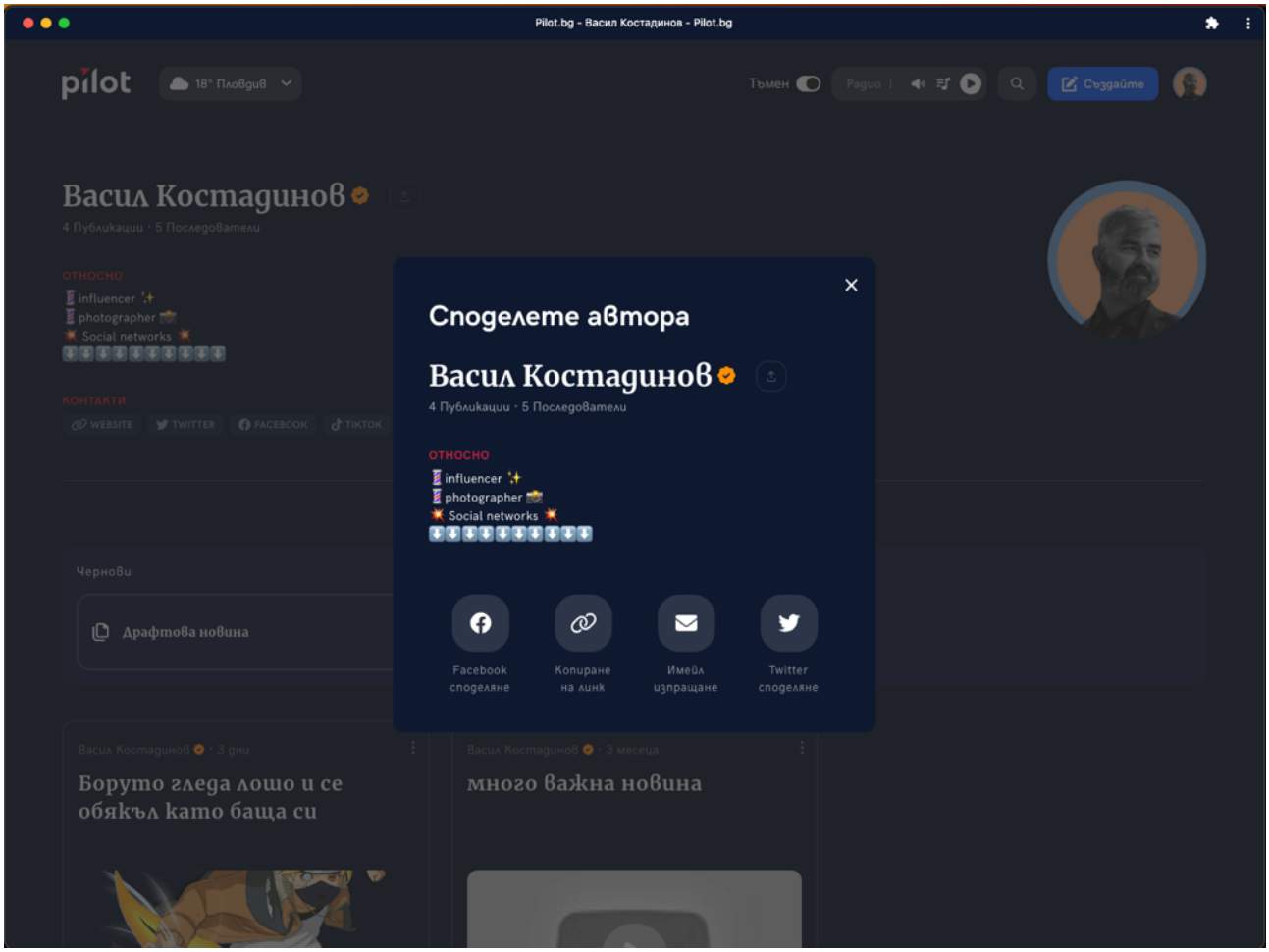
При навигация до потребителския профил се зарежда горната страница. На първо място фигурира името на потребителя, заедно с броя постове, броя последователи и аватара. При липса на име се зарежда служебно генерираното име на потребителя, което е конкатенация на думата “user” с низ от цифри - timestamp (произнася се “таймстамп”) на точното време, в което неговият профил е бил създаден. Така се избягва дублаж на потребителски имена без да се имплементира сложен алгоритъм. Лоша черта на този метод е, че няма да работи коректно след 3ч. 14мин. и 7 секунди на 19 януари 2038г., когато часовникът ще превърти 32 битовата си стойност и ще започне да тече отначало - 1 януари 1970г. Добра черта е, че най-вероятно няма да има регистрирани потребители през този период.

При липса на аватар, ще се зареди служебно генериран такъв, който ще използва потребителското име за референция и ще създаде уникален аватар. За аватарите е използвана библиотеката “Boring Avatars” и нейният “Bauhaus” алгоритъм.

Под главните данни за профила се намира описанието на протребителя. Неговата дължина е ограничена до 150 символа, но може да съдържа символ за нов ред. При липса на описание, секцията се скрива.

Най-отдолу на заглавната секция се намират линкове към социалните медии на потребителя, които се добавят от страницата за настройки. При неналичние на поне една социална мрежа секцията се скрива.

На различно място в разпределението на страницата се намира бутон за споделяне на профила, който отваря модал с малък преглед на страницата и 4 начина за споделяне.



Фиг. 47 Модал за споделяне на потребителски профил

Наподобява модала за споделяне на новина.

В долната секция се помещават 3 компонента в следния ред:

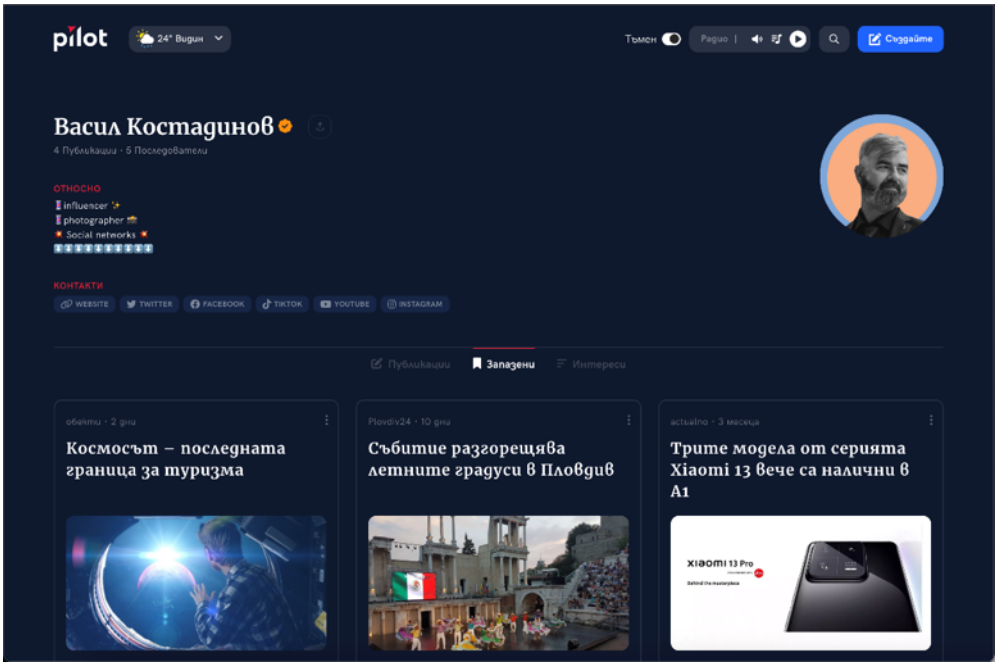
• навигация за 3-те секции на профила, състояща се от иконки и текст в настолната версия

• секция с чернови новини, които при клик изпращат потребителя в редактора, а при липса на чернови секцията се скрива

• секция с вече публикувани новини, като при липса на такива се появява надпис с текст “Няма публикувани новини”

Личен потребителски профил - секция “Запазени новини”

Фиг. 48 Запазени новини - мобилно приложение



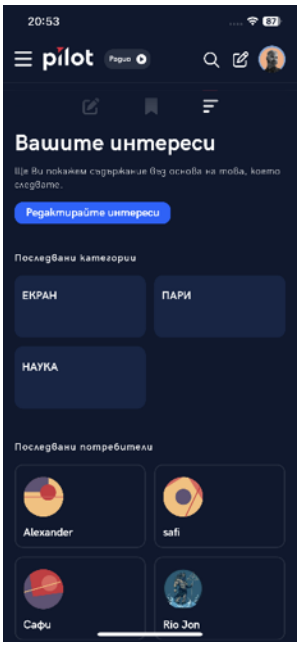
Фиг. 49 Запазени новини - настолно приложение

При навигиране от popover-а с личните данни или при натискане на бутона “Запазени” в потребителския профил се отваря страницата със запазените новини. Тук се зареждат всички статии, запазвани някога чрез този профил в хронологичен ред.

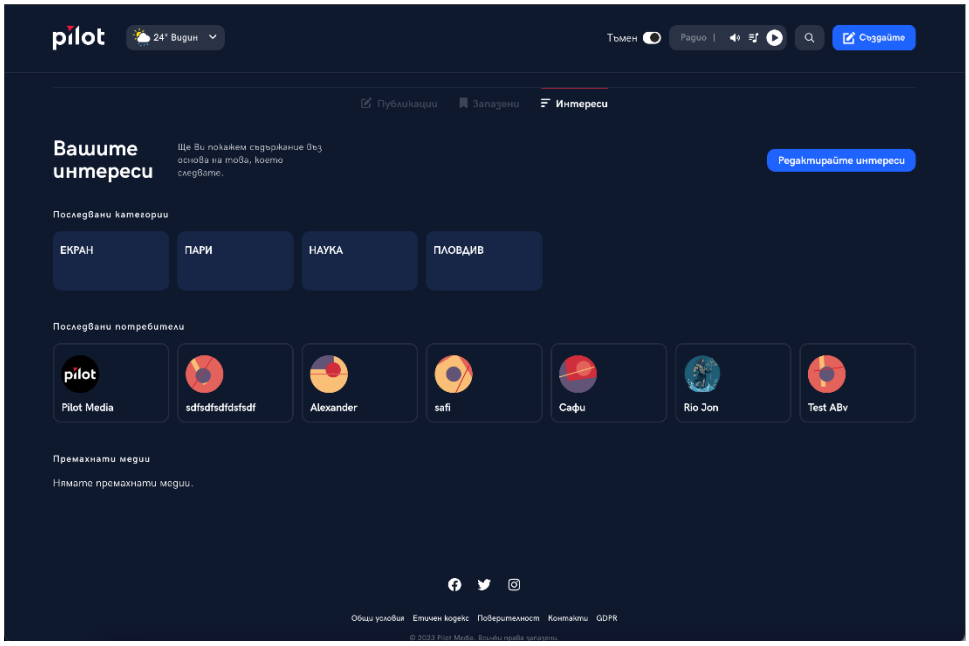
При кликане на новина от списъка, функционалността се запазва. При натискане на запълнената отметка “Bookmark”, новината се маркира като “забравена”, но се се премахва от списъка, в случай, че потребителят е кликнал случайно и/или иска да я запази наново. При повторно зареждане на страницата, новината изчезва от списъка.

При празен списък имаме съответният за това индикатор и текст “Нямате запазени статии”.

Личен потребителски профил - секция “Интереси”



Фиг. 50 Интереси - мобилно приложение



Фиг 51. Интереси - настолно приложение

При навигиране от popover-а с личните данни или при натискане на бутона “Интереси” в потребителския профил се зарежда страницата с интересите на потребителя. Тя представлява 3 списъка, всеки от които е отделна настройка на неговия feed.

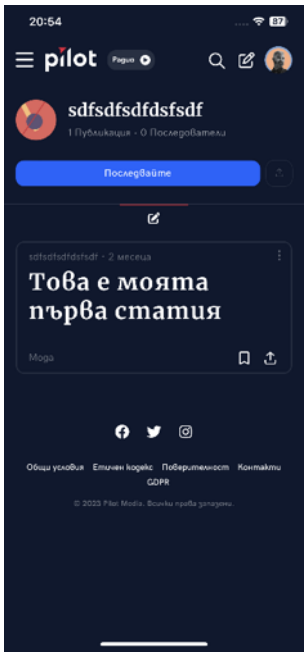
Първата секция е поредица от категориите, които е последвал. При клик върху тях се отива на съответната страница за детайлен преглед на категория или подкатегория.

Втората секция обединява последваните медии и потребители. Тъй като и двата типа потребители пускат новинарско съдържание, те могат да бъдат сложени под един знаменател.

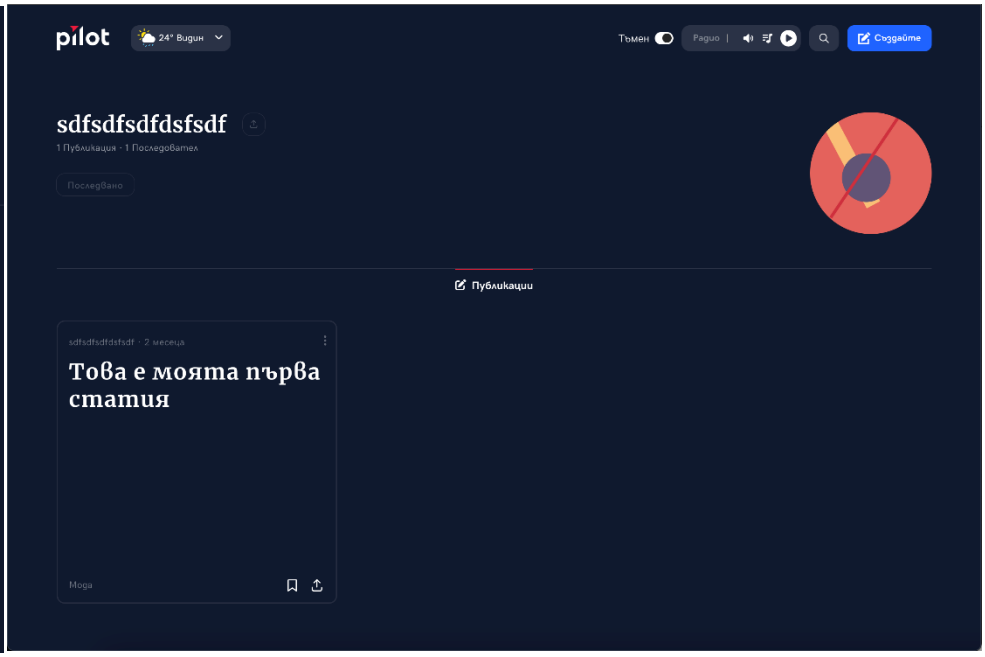
Третата секция показва медиите, които сме блокирали. Всяка новина съдържа в себе си бутон за блокиране на медията. След като една медия е блокирана, тя ще фигурира в този списък и няма да се показва в персоналния feed, независимо колко публикации е направила тя в последваните категории.

При желание за редакция, потребителя може да кликне върху бутона “Редактирайте интереси”, при което ще се отвори гореописаният модал за следване на поне 3 категории.

Публичен профил на друг потребител



Фиг. 52 Публичен профил - мобилно

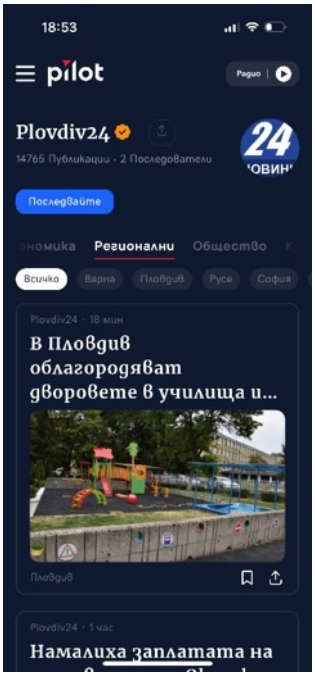


Фиг. 53 Публичен профил - настолно приложение

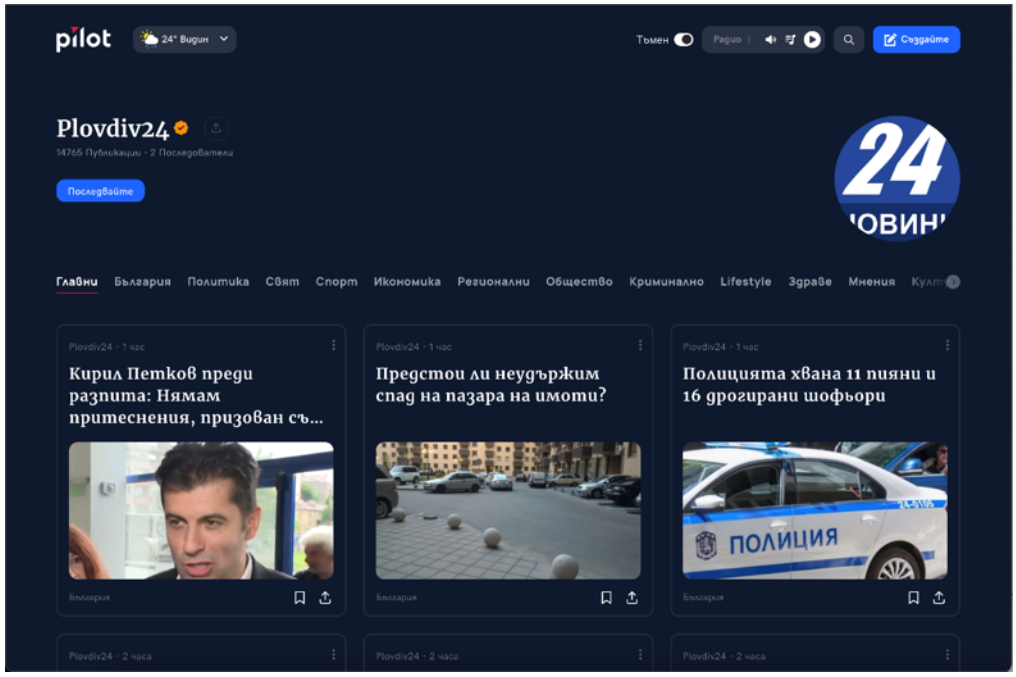
При натискане на който и да е хиперлинк с името на потребител, приложението ще ни пренасочи към страницата за детайлен преглед на публичен профил. Тя наподобява много страницата за личен профил, но липсват съответните бутони за “Запазени” и “Интереси”, както и бутонът за редакция. Не може да се виждат интересите или запазените новини на друг потребител, те са лично негови.

В заглавната част фигурира нов бутон с текст “Последвайте”, който позволява този потребител да бъде добавен към списъка със следвани медии и потребители. При натискане се изпраща заявка, който създава нов запис с две полета: from и to, релации с потребител.

Публичен профил на медия



Фиг. 54 Профил на медия - мобилно приложение



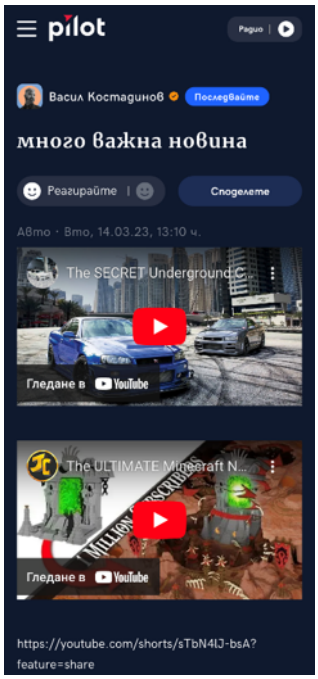
Фиг. 55 Профил на медия - настолно приложение

Публичният профил на медия може да бъде достъпен чрез натискане на името на медията върху всяка новина. Той е почти идентичен с другите два вида профили, с разликата, че винаги е потвърден и притежава сложния филтър по категория от началната страница на нерегистриран потребител.

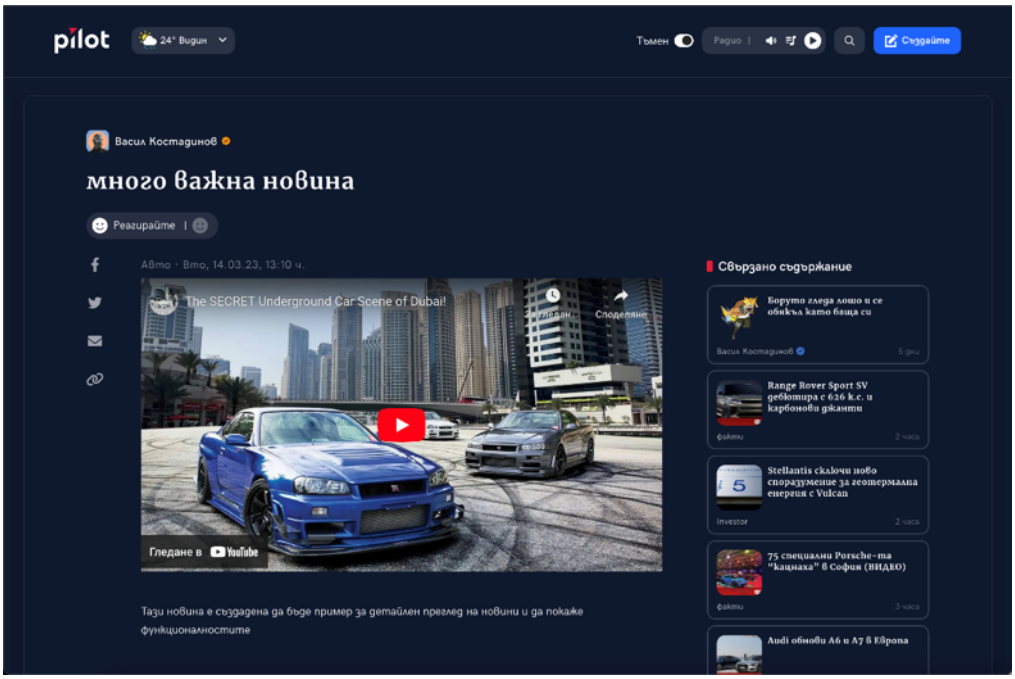
При клик на която и да е категория, процесът е аналогичен с този в страниците на категории и подкатегории - филтрират се новини само от избраните.

Профилът на медия също може да бъде последван и да излиза във персоналния feed на потребителя, дори той/тя да не следва категориите, в които публикува дадената медия.

Детайлен преглед на вътрешна новина



Фиг. 56 Преглед на новина - мобилно приложение



Фиг. 57 Преглед на новина - настолно приложение

При клик върху новина, която не е публикувана от медия се зарежда страницата за детайлен преглед на авторска новина. За разлика от външните новини, принадлежащи на създалите ги медии и отварящи се в нов прозорец, вътрешната новина се зарежда изцяло вътре в приложението, което оптимизира доста процеса и подобрява потребителското изживяване.

В началото на страницата са позиционирани името и аватара на автора, които сами по себе си са хиперлинк към неговия публичен профил. Непосредствено до името се намира бърз бутон за последване. Ако профилът е потвърден, до името се появява оранжево тикче, знак за верификация и доверен източник.

След името на автора, разбира се, е заглавието на статията. То е най-големият надпис на страницата, изписан в таг <h1/>, което допринася за SEO оптимизацията на сайта.

Под заглавието се намира бутона за реакции. При налагане на курсора или задържане под мобилно устройство се отваря списък с възможни реакции. При клик на някоя от тях се изпраща заявка и се създава нов запис в таблицата с реакции, в който има тип на реакцията, релация към потребител, релация към статия и IP address (полето за ID, дата на създаване, дата на публикуване и дата на реакция се създават по подразбиране към всеки един обект в базата данни). Ако реакция вече съществува и потребителят реши да смени типа и, то се изпраща

заявка за редакция на реакцията, като се подава нейното ID и новия тип (like, love, laugh, wow, anger)

При опит за реагиране без регистрация се отваря модала за входиране.

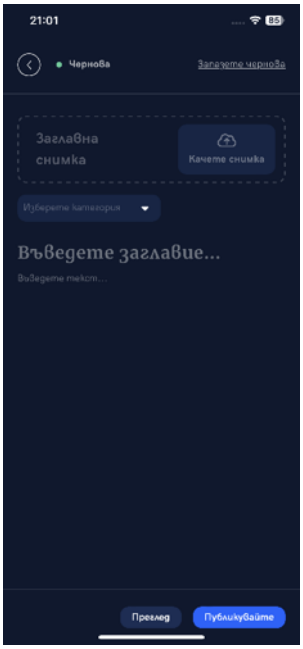
По-надолу следват:

- категорията, към която принадлежи новината - името ѝ е хиперлинк към страницата за детайлен преглед на категория или подкатегория
- 4 бутона за споделяне на статията - отварят модала за споделяне на статия
- часът, денят и датата на публикуване на самата статия
- заглавна снимка (ако има такава, при наличие на видео като първи елемент, заглавната снимка се заменя от player на видеото)
- съдържание на статията - текст, код, цитати, списъци, вградени видеа и други
- заключителен текст, гласящ, че авторът може да бъде подкрепен на платформа от трета страна (стига авторът да е поставил съответния линк в настройките на своя профил)

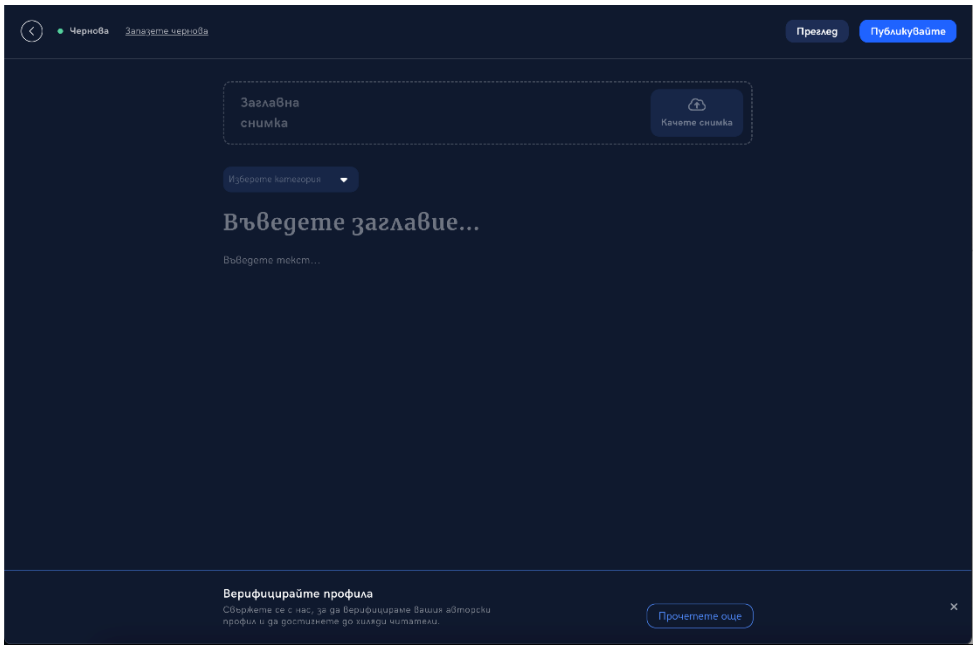
До и непосредствено след съдържанието на статията се помещават карти с линкове към други новини. Статиите от дясно могат, но не е задължително да съдържат други новини на автора и/или новини от същата подкатегория (ако има такава). Новините под статията не съдържат новини на автора, а само статии от същата главна категория.

Накрая детайлния преглед завършва с линкове към легалните страници и към социалните мрежи на приложението.

Екран за създаване на статия



Фиг. 58 Създаване на новина - мобилно приложение



Фиг. 59 Създаване на новина - настолно приложение

При клик върху бутона “Създайте” се зарежда редакторът за създаване на статия. Той е направен така, че максимално да улесни писането и създаването на новини, дори и под мобилно устройство. За целта се използва rich text redactor от библиотеката Editor.js, която предоставя функционалности, подобни на тези на Microsoft word, Google sheets или Apple Pages, а именно:

- редактиране цвета, наклонението и размера на реда и междуредието
- определяне на параграф като заглавие, подзаглавие, цитат и/или др.
- вкарване на блок от програмен код
- създаване на списъци с точки, числа, римски числа или други фигури
- вкарване на снимки, фигури и/или анимации
- интегриране на видеоклипове от платформи като YouTube или Vimeo чрез поставяне на връзка към съответния ресурс
- интегриране на социални постове от Instagram, Facebook, Twitter и др. чрез поставяне на връзка към съответния ресурс
- създаване на хиперлинкове към други уебстраници
- подравняване на параграфите вляво, вдясно или център
- създаване на таблици и много други функционалности.

В началото на страницата се намира карето за качване на снимка. Може да бъде качена една заглавна снимка на статията, която ще се показва на всички размери карти, водещи към нейното съдържание. При натискане на бутона “Качете” се отваря системния диалогов прозорец за избиране на изображение. Графика може да бъде добавена и чрез провлачване под подходящо устройство. След избор, снимката не се качва веднага, а се изчаква процеса по публикуване. Ако новината не бъде публикувана, снимката се изтрива от буферната памет. След публикуване се изпраща заявка към Strapi, който от своя страна предава бинарния файл на Amazon S3, където всъщност се съхраняват снимките. След запазване в S3 се връща линк за файла към Strapi, който го записва в своята таблица Upload. След успешен запис, референция към файла се връща към приложението, за да може процеса по създаване на новина да продължи.

Непосредствено по карето за изображение се намира полето за въвеждане на заглавие към статия. То използва същия шрифт и дължина на полето като при преглед на новина, за да се види точно как ще изглежда и след като новината бъде публикувана.

Под заглавието се намира select компонент с всички категории. При клик се извежда списък, като под компютър дори е възможно да се извърши търсене и филтриране по име.

След като се избере категория, може да се напише съдържание на статията, като то не бива да бъде по-кратко от 20 символа. Вградени материали и/или изображения не допринасят за броя символи.

След като потребителят привърши с редактирането на статията си, той може да натисне бутона “Преглед” в горния десен ъгъл под настолно устройство или в долната част под мобилно устройство. Зарежда се изглед, подобен на този на публикувана новина, като изключение правят свързаните новини отдясно и под статията. Чрез този преглед потребителят може да се увери, че неговата статия изглежда добре и е четима под всяко устройство и, че всички графики и/или интеграции се визуализират правилно.

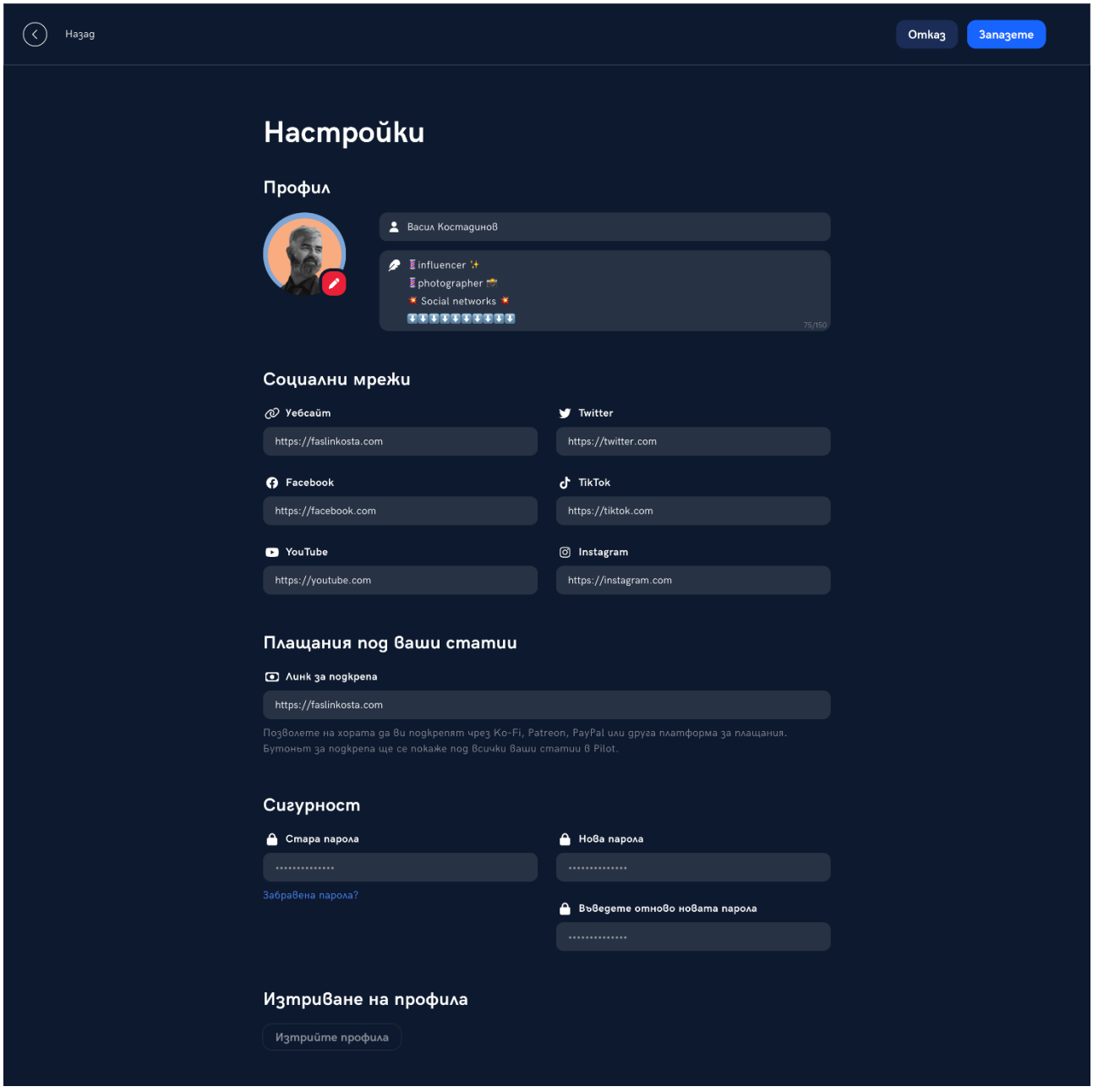
При положително впечатление от новината, тя може да бъде публикувана, чрез натискане на бутона “Публикувайте”

Ако новината не се харесва достатъчно, може да бъде натиснат бутона “Запази като чернова”, тогава ще бъде зададен статус draft на статията и тя няма да бъде публикувана, но ще бъде запазена в базата данни, където ще може да бъде редактирана по всяко време.

Ако новината не е достатъчно добра и потребителят реши, че не иска да я публикува или запазва, може да натисне бутона назад. Ще се покаже прозорец, констатиращ, че прогресът не е запазен.

При опит за публикация на невалидна новина ще се появи червена грешка с надпис, указващ нередността. След нейното поправяне, новината може да бъде публикувана, прегледана или запазена като чернова.

Настройки на профил



Фиг. 60 Настройки на профил

При кликане на бутона “Настройки” се зарежда екран с няколко полета, съдържащи данните на потребителя. При първоначална регистрация всички полета, включително и карето за аватар, ще бъдат празни.

Полетата са както следва:

• каре за качване на профила снимка - аватар - ще се показва над написаните от потребителя статии

• текстово поле за име и фамилия - ще се показва над и под статиите на потребителя

• Поле за описание - ще се показва само в потребителския профил, като то може да бъде дълго максимум 150 символа

• 6 полета за най-известните социални мрежи - потребителят може да остави своите връзки тук, които ще се появят на неговия потребителски профил

• линк за спонсорство - ако потребителят притежава акаунт в сайт на трета страна, който позволява приема на дарения (пр. Patreon, LinkTree и др.) може да остави линка си в това поле, а при наличието на линк ще се показва допълнителен параграф под всяка негова статия с указания как може да се дарят средства

• 3 полета за смяна на паролата - стандартни полета, едното от които е за старата парола и две за новата, като под първото поле се помещава линк “Забравена парола”, който изпраща имейл до потребителя с код за валидация и линк за смяна на паролата

• бутон за изтриване - ако потребителят не желае да съществува повече в мрежата на приложението, то той може да изтрие своя профил, с което се изтрива всяка негова новина, реакция и други доказателства за неговото съществуване и това действие е окончателно и следователно не може да бъде отменено.

Заключение

В съвременното ежедневие дезинформацията е неоспорим факт. Всеки потребител, притежаващ smart устройство като телефон, настолен компютър или лаптоп с връзка към глобалната мрежа може да създаде и разпространи невярна информация, която да достигне хиляди други ползватели на интернет.

За подобряване качеството на информационното разпространение, много доверени източници създават свои канали с добре познати имена и домейни, които трудно могат да бъдат фалшифицирани. Някои по-големи компании, като Google, създават така наречените агрегатори, които да извличат новини само от места, които те смятат за сигурни, увеличавайки доверието на потребителите си.

На пръв поглед, функцията на тези агрегатори е доста полезна, поради простия факт, че не трябва да се търсят новини на различни места. В забързаното ежедневие времето е ценно, а получаването на точната информация в точното време - още по-ценно. При проучване на пазара в България, обаче, се забелязва липсата на подобен прост и лесен за употреба агрегатор на новини. Поради тази причина е създаден този софтуерен продукт - приложение за събиране и филтриране на новини от всички надеждни родни източници и изцяло на български език.

Многоплатформеното приложение предоставя на потребителя лесен достъп до неговите любими източници, без да се налага той да събира информацията от тях поотделно. То може да се инсталира лесно на всяко едно устройство, независимо дали неговата система е Android, iOS, Windows, Linux или друга, като се поддържат различни размери на разположението и разпределението за всяка големина устройства. При желание да се създаде списък от любими източници, регистрацията става чрез попълване на две полета - име и парола, или чрез социалната мрежа Facebook, която най-вероятно вече е инсталирана и оторизирана на потребителското устройство. При вече налична регистрация, потребителят може да се наслади на един безкраен списък от новини, който трудно може да бъде изчерпан. А ако ползвателят има мнение за изказване, може да го направи на собствения си профил.

Текущите функционалности правят приложението лесно за ползване, просто за научаване и бързо за навигиране. Резултат може да бъде намерен в рамките на секунди, независимо каква тема бива потърсена, стига тя да фигурира в някой от българските източници.

Бъдещите планирани функционалности са, но не се ограничават до: Добавяне на коментари и отговори, реакции на коментари, онлайн статус, чат, поддръжка на native функции, известия.

Използвана литература

1. “JavaScript професионални проекти”, Джереми МакПийк, Пол Уилтън, DuoDesign, 2005, ISBN 9548396162
2. „JavaScript For Dummies”, Ричард Уогнър, АлексСофт, 2011. ISBN 9546562319
3. “Modern Full-Stack Development”, Frank Zammetti, Apress, 2020, ISBN 1484257375
4. “HTML & CSS: Добрите страни”, Бен Хеник, ЗеСТ Прес, 2011, ISBN 9549341355
5. “React and React Native”, Roy Derks, Adam Boduch, Packt Publishing, 2020, ISBN 1839211148
6. https://react.dev/reference/react (юли 2023)
7. https://nodejs.org/en/about (юли 2023)
8. https://www.apollographql.com/docs/react (юли 2023)
9. https://www.meilisearch.com/docs (юли 2023)
10. https://docs.strapi.io/ (юли 2023)
11. https://www.typescriptlang.org/docs/ (юли 2023)
12. https://www.javascript.com/resources (юли 2023)
13. https://nextjs.org/docs (юли 2023)
14. https://vercel.com/docs (юли 2023)
15. https://www.postgresql.org/docs/ (юли 2023)
16. https://web.dev/progressive-web-apps/ (юли 2023)
17. https://docs.pwabuilder.com/#/ (юли 2023)
18. https://docs.staruml.io/ (юли 2023)
19. https://news.google.com/home (юли 2023)
20. https://flipboard.com/ (юли 2023)
21. https://css-tricks.com/ (юли 2023)
22. https://docs.digitalocean.com/ (юли 2023)
23. https://fontawesome.com/docs (юли 2023)
24. https://openweathermap.org/api/one-call-3 (юли 2023)
25. https://docs.npmjs.com/ (юли 2023)
26. https://yarnpkg.com/ (юли 2023)
27. https://iterm2.com/ (юли 2023)
28. https://github.com/ohmyzsh/ohmyzsh/wiki (юли 2023)
29. https://stackoverflow.com/ (юли 2023)
30. https://stackexchange.com/ (юли 2023)

31. https://github.com/community (юли 2023)
32. https://developers.facebook.com/docs/facebook-login/ (юли 2023)
33. https://code.visualstudio.com/docs (юли 2023)
34. https://www.w3schools.com/ (юли 2023)
35. https://www.youtube.com/kevinpowell (юли 2023)
36. https://www.youtube.com/c/Hyperplexed (юли 2023)
37. https://www.youtube.com/@WebDevSimplified (юли 2023)