# ВВЕДЕНИЕ

Книгопечатная продукция утрачивает свои позиции на мировом рынке и к 2019 году число книг, проданных в электронном формате, сравняется с продажами бумажных экземпляров, а потом и неминуемо обгонит их. Это обусловлено широким распространением смартфонов, планшетов и стоимостью электронных книг, которые как правило дешевле печатных изданий, так как очевидно, не требуют материалов для производства каждого экземпляра.

В настоящий момент существует большое количество решений для чтения электронных книг, большая часть которых состоит в программном обеспечении, устанавливаемом на устройстве пользователя. Приложения на мобильных телефонах, планшетах и компьютерах имеют ряд преимуществ, таких как скорость работы, использовании библиотеки пользователя и широкие возможности для разработчиков.

Недостатком описанных выше приложений является привязка пользователя к одному устройству, на котором он обычно читает. При переходе, например, с мобильного телефона на компьютер или планшет, пользователь вынужден использовать другое ПО, загружать книгу и тратить время на поиск места, где чтение было прервано, что занимает не так мало времени.

Благодаря последним стандартам представления данных в сети интернет, таким как HTML5, CSS3, веб-приложения могут приблизиться по функциональности и скорости работы к нативным приложениям. Веб-приложение может решить проблему синхронизации при использовании разных устройств для чтения, предоставляя единый интерфейс для любых устройств, учитывая при этом особенности управления и разное разрешение экранов. Другим преимуществом веб-приложений можно считать возможность разработчика вносить изменение в приложение по функционалу, стабильности работы, внешнего вида и любые другие, без необходимости в обновлении ПО пользователем.

Целью дипломной работы является разработка веб-приложения для чтения электронных книг на любых современных устройствах, которые для этого подходят. Благодаря приложению доступ ко всем загруженным книгам может быть осуществлен с любого устройства имеющего доступ в интернет.

Для разработки программного обеспечения, которое бы учитывало описанные недостатки существующих решений, необходимо:

* провести обзор веб-приложений, которые предоставляют интерфейс для чтения книг;
* провести обзор популярных форматов электронных книг;
* провести обзор документации выбранных форматов;
* разработать структуру базы данных;
* разработать UI-компоненты веб-интерфейса;
* определиться с архитектурой клиентской части веб-приложения;
* написать бизнес логику клиентской части веб-приложения;
* определиться с архитектурой серверной части веб-приложения;
* написать бизнес логику серверной части веб-приложения.

# АНАЛИЗ

# Анализ и описание предметной области

В связи с тем, что уже в текущее время люди имеют не одно, а несколько устройств, для чтения они могут выбирать устройство в зависимости от ситуации. Например, в метро, где в час пик недостаточно пространства для планшета или электронной книги, самым распространённым устройством является смартфон. На работе или дома удобнее пользоваться компьютером или планшетом. При смене устройств возникает необходимость иметь копии книг на каждом устройстве.

Суть разрабатываемого приложения состоит в том, что бы пользователь имел доступ к большому количеству книг с любого устройства на котором присутствует браузер и доступ в интернет. Приложение должно иметь собственный каталог с книгами из которых пользователь может выбрать любую, добавить себе в библиотеку и начать читать, автоматически должна сохраняться информация о количестве прочитанных страниц, чтобы при повторном открытии книги она открывалась с того места где пользователь остановился в прошлый раз. Часть книг должна быть доступна только по подписке, а часть бесплатна. Пользователь должен иметь возможность оформить подписку на разные периоды (1 месяц, 3 месяца, 6 месяцев). Так же пользователь должен иметь возможность загрузить собственные книги в формате epub.

# Обзор аналогичных веб-приложений

На рисунке 2 представлено веб-приложение EpubReader. Приложение довольно простое в использовании. Оно не имеет своей библиотеки, но позволяет пользователю загружать любую книгу и читать её онлайн.

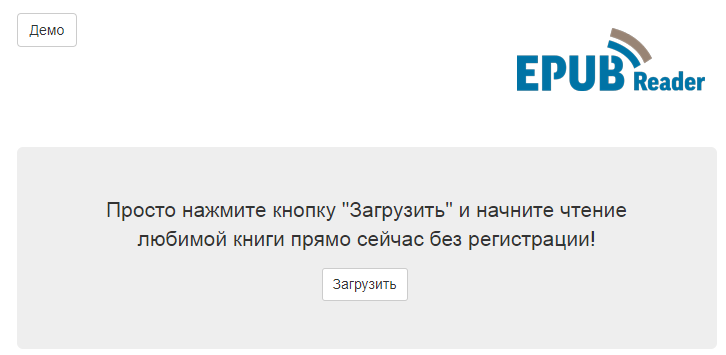


Рисунок 1 – EpubReader

На рисунке 3 представлено веб-приложение bookmate. Данное приложение имеет так же загружаемую на устройство версию. В приложении нет возможности загружать собственные книги, но есть огромная библиотека свыше 65000 наименований

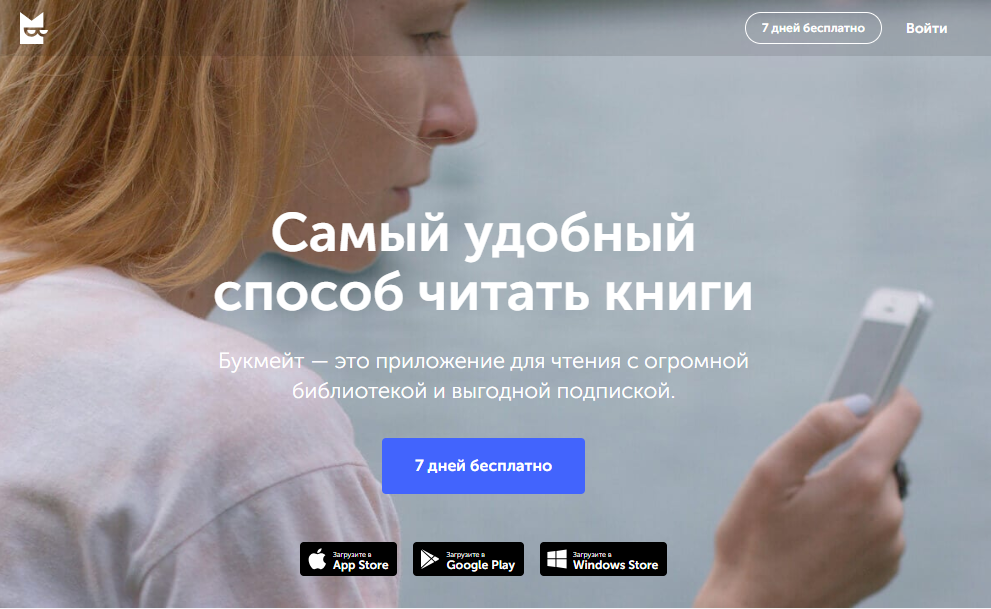


Рисунок 3 – bookmate

# Обзор технологий разработки веб-приложений

На сегодняшний день существует два принципиальных подхода к созданию веб-приложений: традиционные веб-приложения, большая часть логики которых выполняется на сервере, а также одностраничные приложения, логика пользовательского интерфейса которых выполняется преимущественно в веб-браузере, а взаимодействие с веб-сервером осуществляется главным образом через веб-API. Также возможен гибридный подход, при котором в простейшем случае в рамках крупного традиционного веб-приложения размещаются одно или несколько полнофункциональных подчиненных приложений, построенных на основе одностраничной модели.

Традиционные веб-приложения обычно используются в следующих случаях:

* На стороне клиента к приложению применяются минимальные требования, например, используются только функции чтения.
* Приложение должно работать в браузерах без поддержки JavaScript.
* Ваша команда не знакома с принципами разработки на JavaScript или TypeScript.

Одностраничные приложения рекомендуется применять в следующих случаях:

* В приложении требуется полнофункциональный пользовательский интерфейс.
* Ваша команда знакома с принципами разработки на JavaScript или TypeScript.
* В вашем приложении должен предоставляться API для других внутренних или общедоступных клиентов.

Кроме того, для работы с платформой одностраничных приложений требуется гораздо больший опыт в построении архитектуры и обеспечении безопасности. Они характеризуются большей частотой обновления и появления новых платформ по сравнению с традиционными веб-приложениями. Кроме того, по сравнению с традиционными веб-приложениями при работе с одностраничными приложениями усложняются настройка процессов автоматизированного построения и развертывания, а также использование таких вариантов развертывания, как контейнеры.

Эти моменты необходимо учитывать, оценивая улучшения модели взаимодействия с пользователем, которые могут дать одностраничные приложения.

# Подходы к написанию Web-приложений

На сегодняшний день существует два принципиальных подхода к созданию веб-приложений: традиционные веб-приложения, большая часть логики которых выполняется на сервере, а также одностраничные приложения, логика пользовательского интерфейса которых выполняется преимущественно в веб-браузере, а взаимодействие с веб-сервером осуществляется главным образом через веб-API. Также возможен гибридный подход, при котором в простейшем случае в рамках крупного традиционного веб-приложения размещаются одно или несколько полнофункциональных подчиненных приложений, построенных на основе одностраничной модели.

Традиционные веб-приложения обычно используются в следующих случаях:

* На стороне клиента к приложению применяются минимальные требования, например, используются только функции чтения.
* Приложение должно работать в браузерах без поддержки JavaScript.
* Ваша команда не знакома с принципами разработки на JavaScript или TypeScript.

Одностраничные приложения рекомендуется применять в следующих случаях:

* В приложении требуется полнофункциональный пользовательский интерфейс.
* Ваша команда знакома с принципами разработки на JavaScript или TypeScript.
* В вашем приложении должен предоставляться API для других внутренних или общедоступных клиентов.

Кроме того, для работы с платформой одностраничных приложений требуется гораздо больший опыт в построении архитектуры и обеспечении безопасности. Они характеризуются большей частотой обновления и появления новых платформ по сравнению с традиционными веб-приложениями. Кроме того, по сравнению с традиционными веб-приложениями при работе с одностраничными приложениями усложняются настройка процессов автоматизированного построения и развертывания, а также использование таких вариантов развертывания, как контейнеры.

Эти моменты необходимо учитывать, оценивая улучшения модели взаимодействия с пользователем, которые могут дать одностраничные приложения.

# Анализ инструментов разработки веб-приложений

Webpack *–* это инструмент, позволяющий скомпилировать, например, JavaScript модули в единый JS-файл. Webpack также известен как сборщик модулей.

При большом количестве файлов он создает один объемный файл (или несколько файлов) для запуска вашего приложения.

Он также способен выполнять множество иных операций:

* помогает собрать воедино ваши ресурсы
* следит за изменениями и повторно выполняет задачи
* может выполнить транспиляцию JavaScript следующего поколения до более старого стандарта JavaScript (ES5) с помощью Babel, что позволит использовать новейшие функции JavaScript, не беспокоясь о том, поддерживает их браузер или нет
* может выполнить транспиляцию CoffeeScript в JavaScript
* может конвертировать встроенные изображения в data:URI
* позволяет использовать require() для CSS файлов
* может запустить webpack-dev-server (в нём встроен локальный сервер и livereload (“живая перезагрузка браузера”))
* может работать с Hot Module Replacement (замена горячего модуля)
* может разделить выходной файл (output file) на несколько файлов, чтобы избежать медленной загрузки страницы из-за большого размера JS-файла
* может выполнить Tree Shaking

Webpack не ограничивается одним лишь фронтендом, его также успешно применяют в бэкенд разработке на Node.js.

[Gulp](http://gulpjs.com/)*–* инструментарий, облегчающий жизнь веб-разработчика, с его помощью можно задавать различные задачи. Часто используется для таких front-end задач как: поднятие сервера, автоматическое обновление страницы браузера в любой момент сохранения файла, использование препроцессоров как Sass или LESS, оптимизация CSS, JavaScript и изображений.

Docker — программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой [контейнеризации](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%B9%D0%BD%D0%B5%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F). Позволяет «упаковать» приложение со всем его [окружением](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BE%D0%BA%D1%80%D1%83%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5&action=edit&redlink=1) и зависимостями в контейнер, который может быть перенесён на любую [Linux](https://ru.wikipedia.org/wiki/Linux)-систему с поддержкой [cgroups](https://ru.wikipedia.org/wiki/Cgroups) в [ядре](https://ru.wikipedia.org/wiki/%D0%AF%D0%B4%D1%80%D0%BE_Linux), а также предоставляет среду по управлению контейнерами. Изначально использовал возможности [LXC](https://ru.wikipedia.org/wiki/LXC), с 2015 года применял собственную библиотеку, абстрагирующую виртуализационные возможности ядра Linux — libcontainer. С появлением ​Open Container Initiative начался переход от монолитной к модульной архитектуре.

Разрабатывается и поддерживается одноимённой компанией-[стартапом](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%80%D1%82%D0%B0%D0%BF), распространяется в двух редакциях — общественной (Community Edition) по [лицензии Apache 2.0](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F_Apache) и для организаций (Enterprise Edition) по [проприетарной](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BF%D1%80%D0%B8%D0%B5%D1%82%D0%B0%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)лицензии. Написан на языке [Go](https://ru.wikipedia.org/wiki/Go_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)).

Visual Studio Code  —  [редактор исходного кода,](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%B8%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BE%D0%B4%D0%B0) разработанный [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) для [Windows](https://ru.wikipedia.org/wiki/Windows), [Linux](https://ru.wikipedia.org/wiki/Linux) и [macOS](https://ru.wikipedia.org/wiki/MacOS). Позиционируется как «лёгкий» редактор кода для [кроссплатформенной](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C) разработки [веб-](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) и [облачных](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BB%D0%B0%D1%87%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F) приложений. Включает в себя [отладчик](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BB%D0%B0%D0%B4%D1%87%D0%B8%D0%BA) инструменты для работы с [Git](https://ru.wikipedia.org/wiki/Git), [подсветку синтаксиса](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D1%81%D0%B2%D0%B5%D1%82%D0%BA%D0%B0_%D1%81%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81%D0%B0),  [IntelliSense](https://ru.wikipedia.org/wiki/IntelliSense) и средства для [рефакторинга](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3). Имеет широкие возможности для [кастомизации](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%81%D1%82%D0%BE%D0%BC%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F): пользовательские темы, [сочетания клавиш](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%87%D0%B5%D1%82%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BA%D0%BB%D0%B0%D0%B2%D0%B8%D1%88) и [файлы конфигурации](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB_%D0%BA%D0%BE%D0%BD%D1%84%D0%B8%D0%B3%D1%83%D1%80%D0%B0%D1%86%D0%B8%D0%B8). Распространяется бесплатно, разрабатывается как программное обеспечение с [открытым исходным кодом](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), но готовые сборки распространяются под [проприетарной лицензией](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BF%D1%80%D0%B8%D0%B5%D1%82%D0%B0%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5).

Visual Studio Code основан на [Electron](https://ru.wikipedia.org/wiki/Electron) — [фреймворк](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA), позволяющий с использованием [Node.js](https://ru.wikipedia.org/wiki/Node.js) разрабатывать настольные приложения, которые работают на [движке Blink](https://ru.wikipedia.org/wiki/Blink_(%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA)). Несмотря на то, что редактор основан на [Electron](https://ru.wikipedia.org/wiki/Electron), он не использует редактор [Atom](https://ru.wikipedia.org/wiki/Atom_(%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%80%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80)). Вместо него реализуется веб-редактор Monaco разработанный для Visual Studio Online.

ESLint — это инструмент, который позволяет проводить анализ качества вашего кода, написанного на любом выбранном стандарте JavaScript. Он приводит код к более-менее единому стилю, помогает избежать глупых ошибок, умеет автоматически исправлять многие из найденных проблем и отлично интегрируется со многими инструментами разработки, такими как редактор Visual Studio Code и Web Storm IDE от JetBrains. ESLint не обязывает вас к одному какому-то конкретному стилю. Наоборот — вы можете выбрать что-то из лучших практик и доработать по своему усмотрению!