**Компьютерная академия «ШАГ»**

**Днепропетровский филиал**

**Кафедра Разработки программного обеспечения**

**ДИПЛОМНЫЙ ПРОЕКТ**

**«BookTr – читалка книг с заучиванием слов»**

Группа: FD18

Студенты:

Загляда И.М.

Научный руководитель:

???

Кривой Рог 2019

Аннотация

Разработан сайт «Перевод книг FB2 с функцией заучивания слов». Сайт предназначен для людей, изучающих иностранные языки. В первую очередь, это конечно же изучение английского языка русскоязычными пользователями. Ключевыми требованием при разработке являлось:

1. *Переложить максимум нагрузки с сервера на сторону клиента (браузер), для обслуживания максимально возможного количества пользователей с минимальными затратами для владельца ресурса.*
2. *Сделать SPA-сайт с одной точкой фхода, после чего не перезагружать страницу.*

Как всем известно, изучать слова скушно и неинтересно, что ведет к демотивации учащихся. Совсем другое дело, когда изучение слов происходит в процессе чтения любимых книг. Именно этой цели и посвящена данная дипломная работа.

Сайт встречает пользователя страницей выбора родного и изучаемого языков (раздел по умолчанию «Выбор языка»). После выбора языков, происходит перелокализация приложения без перезагрузки страницы (AJAX).

Далее пользователь попадает в раздел «Библиотека». В этом разделе предоставляется возможность выбрать книгу формата FB2 с локального диска. Выбранная книга преобразовывается во внутренний формат, сохраняется во внутренней БД (IndexedDB), выводится в компонент «Чтение» куда и перенаправляется пользователь .

Находясь в разделе «Чтение», пользователь может читать книгу. Клик на слове выводит модальное окно с переводом и предложением изучить слово, либо добавить его в список «Изученные» . Изученные слова имеют прозрачный фон, выбранные для изучения помечаются желным цвепом, неизвестные светло-кирпичным цветом. Чем больше слов знает пользователей, тем меньше будет помеченных слов.

В любой момент пользователь может перейти в раздел «изучение слов» кликнув анимированное меню в правом верхнем меню экрана. Тут его встретит грид-компонент, содержащий все встреченные слова. Компонент обладает функцией постраничного отображения списка с возможностью быстрого перехода на любую страницу. Однако главной функцией раздела является изучение слов, перейти к которому можно нажав на кнопку «Начать изучение». Слова изучаются в прямом и обратном направлениях, после чего сохраняются в БД и перемаркировывается раздел «Чтение». Статистика всегда отображается в верхнем левом углу страницы.

Содержание

1. Введение 4
2. Техническое задание 5
3. Выбор технологии реализации проекта 7
4. Разработка сайта ... 8
   1. Разработка структуры системы 8
   2. Разработка базы данных 10
   3. Структура программного проекта 12
   4. Разработка интерфейса 13
   5. Руководство пользователя 15
5. Выводы 26
6. Список использованной литературы 27

* Введение

**Зачем изучать иностранные языки?**

* Если говорить об английском языке, то он де-факто является международным языком, на котором Вас поймут в любой стране Европы, в Австралиии Америке. Во многих странах Азии и Африки. Вы можете объехать полмира владея одним лишь языком. Особенно ценен он для сферы IT, ведь вся самая свежая информация доступна на английском языке, самые денежные заказчики общаются на английском, да и просто промониторив рынок труда можно убедиться, что без английского языка никак не обойтись.
* Если говорить о других языках, то тут у каждого своя мотивция. Кто-то мечиает об эммиграции в Германию, либо Италию, кто-то желает прочитать книгу на языке оригинала, а кто-то просто расширяет свой кругозор.

2. Техническое задание

**Описание:**

«BookTr – перевод книг» – сайт для широкой аудитории желающих пополнить свой словарный запас. Пользователи читают книги, которые сами же загружают в свою библиотеку. По мере чтения добавляют слова в список «Для изучения». В любой удобный момент пользователь может перейти к заучиванию слов.

Перед разработчиком ставятся следующие технические требования:

1. Минимальная цена разработки, развертывания и содержания.
   1. Сайт должен вызывать минимальную нагрузку на сервер и быть неприхотливым к хостингу.
   2. Вся работа с контентом (книги) должна производиться на стороне пользователя. Не допускается загрузка книг на сервер. При этом после перезагрузки браузера не должна теряться библиотека пользователя и список заученных слов.
2. Сайт должен работать без перезагрузки страниц, т.е. быть классическим SPA-приложением.
3. Требования к браузеру не предъявляются. Допустимо использовать функционал самых современных браузеров Mozilla Firefox и Google Chrome.

**Список действий:**

1. Создать и клонировать репозиторий на GitHub.
2. Проектирования базы данных.
3. Создание базы данных.
4. Вывод информации на макет.
5. Глобальное тестирование и завершение всего проекта.

3. Выбор технологии для реализации проекта

Для создания сайта были выбраны следующие технологии:

* язык JavaScript
* язык разметки гипертекста HTML (HTML5)
* каскадные таблицы стилей (CSS3)
* технология обращения к серверу без перезагрузки страницы Ajax
* Google-translate – сервис, позволяющий получить перевод слов.
* IndexedDB – БД для хранения книг на стороне браузера.
* Promises – способ организации асинхронного кода — объект, представляющий результат успешного или неудачного завершения асинхронной операции.

Детальнее о выборе технологий:

* Язык JavaScript.   
  Это родной для всех браузеров скриптовый язык. Исходя из п.1 технического задания, пользуемся принципом «Быстро пишем прототим, а если взлетит, то переписываем прототип на более подходящем языке, типа TypeScript».
* HTML5, CSS3 – без них данный сайт не напишешь.
* Ajax – диктуется п.2 технического задания и необходимостью работать с Google Translate API.
* Google-translate – сервис, позволяющий получить перевод слов. Избавляет нас от необходимости составлять и поддерживать собственные словари, что весьма накладно. Перенос запросов на сторону браузера позволяет также издавиться от затрат, связанных с политикой Google на тарификацию за переводы.  
  Goole позволяет получить перевод фразы вызывав следующий запрос. https://translate.googleapis.com/translate\_a/single?client=gtx&sl=auto&dt=t&dt=bd&dj=1&text={{some%20tex}}t&tl=ru
* IndexedDB – IndexedDB – это современное NoSQL хранилище данные в формате JSON внутри браузера.   
  Проанализировав технологии, доступные на стороне браузера, приходим к выводу что альтернативы нет. Каждая книга формата FB22 занимает от 2 до 20МБ (в зависимости от количества вложенных изображений ). Это исключает возможность истользования Cookies (до 4х КБ) и Local Storage (обычно до 5ти МБ).   
  WebSQL использовать удобнее IndexedDB, но консорциум W3C прекратил работу над спецификацией в ноябре 2010 года, а это означает что не исключен отказ от поддержки технологии разработчиками тех браузеров, которые успели ее внедрить.
* Promises – Это буквально панацея от callback hall, который особенно ярко выражен при работе с IndexedDB. Без них код очень быстро становится запутанным и трудно поддерживаемым.

4. Разработка сайта

4.1. Разработка структуры системы

При разработке принято решение весь код организовать в виде псевдостатических классов JavaScript, которые относительно легко могут быть преобразованы в стандартные классы(если возникнет такая необходимость). Програмный код разделен на две крупных части, каждая из которых делится на мелкие составляющие. Составляющие приложения выстроены в иерархическую структуру, в которой крупные компоненты являются контейнерами бля меньших.

1. App – дерево статических классов, которые обслуживают непосредственно приложение. Это сам класс App – центральная точка входа и такие компоненты, как навигация, модалные окна, таблицы, адаптер БД, классы исключений, адаптер локализации.
2. Helper – дерево статических классов, которые являются библиотеками-помошниками. Это автономные единицы, которые могут без изменений переиспользоваться в других проектах. Сюда входят такие модули, как “Hash” для выполнения хеширования текста; “Google” помошник для работы с сервисами Google; “Xml” – для работы с данными формата XML; и т.д.

На Рис.1 можно в ощих чертах ознакомиться со структурного кода. Назначение и детальное описание наиболее значимых частей будет сделано в процессе описания функционала.

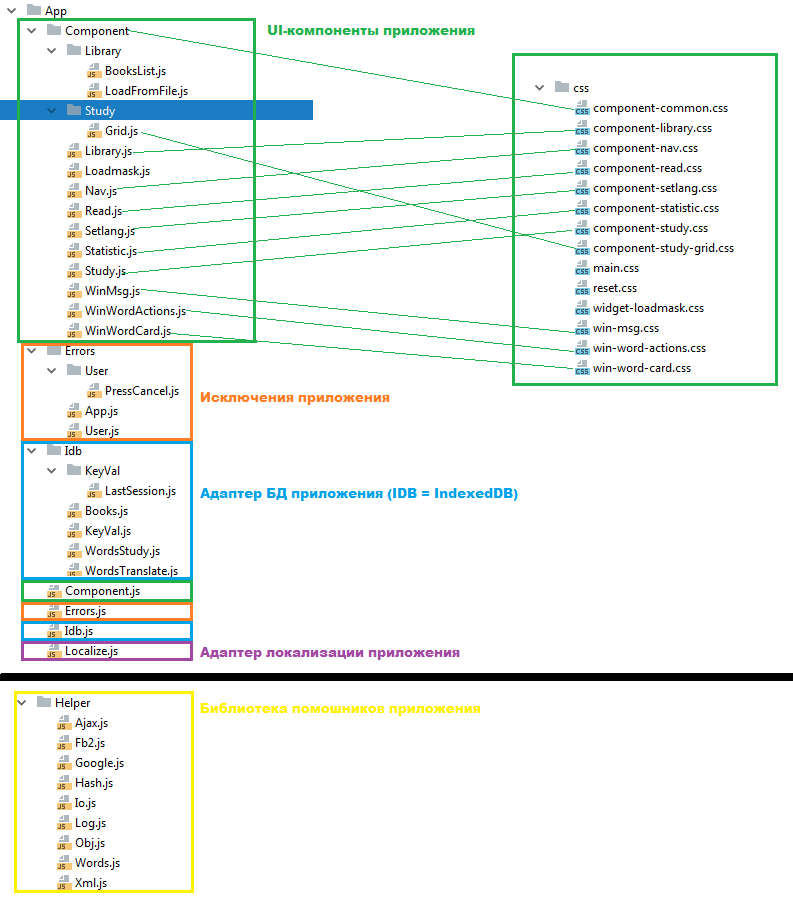
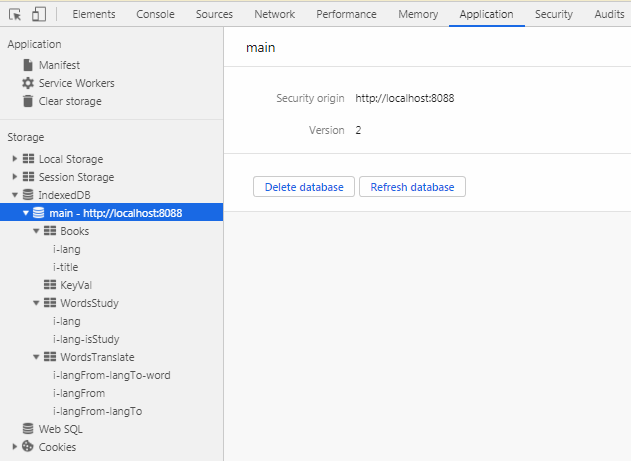


Рис.1 Структура програмного кода приложения.

4.2. Разработка БД приложения.

Общая структура БД приложения представлена на рис.2.

* “main” – имя БД  
    
  Рис.2 БД IndexedDB в панели «Инструменты разработчика» Google Chrome

Как видно, для решения поставленной задачи достаточно четырех таблиц.

Создание и обновление БД инкапсулировано в «\app\js\App\Idb.js» (См.ниже). Алгоритм обновления заключается в последовательном выполнении миграции с версии 0 на 1, с 1 на 2 и т.д. Такой подход дает возможность сконцентрироваться на разнице в структуре соседних версий и выполнять обновление с меньшим шансом ошибки.

*// \app\js\App\Idb.js*  
onUpgradeNeeded(event) {  
 **let** tmpObjStore;  
 **let** transaction = event.**target**.transaction;  
 */\*\*  
 \*  
 \** ***@type*** *{IDBDatabase}  
 \*/* **let** db = event.**target**.result;  
  
 **switch** (event.oldVersion) {  
 **case** 0 :  
 *//-------------------------------------------------------------  
 // Таблица "Хеш-таблица"* tmpObjStore = db.createObjectStore(**'KeyVal'**);  
 *//-------------------------------------------------------------  
 // Таблица "Сохраненные книги"* tmpObjStore = db.createObjectStore(**'Books'**, {  
 **keyPath**: **'hash'**,  
 **autoIncrement**: **false** });  
 tmpObjStore.createIndex(**'i-lang'**, **'lang'**);  
 *// tmpObjStore.createIndex('i-hash', 'hash', {  
 // unique: true  
 // });* tmpObjStore.createIndex(**'i-title'**, **'title'**, {  
 **unique**: **false** });  
 *//-------------------------------------------------------------  
 // Таблица "Изучаемые слова"* tmpObjStore = db.createObjectStore(**'WordsStudy'**, {  
 **keyPath**: [**'lang'**, **'word'**],  
 **autoIncrement**: **false** });  
 tmpObjStore = transaction.objectStore(**'WordsStudy'**);  
 tmpObjStore.createIndex(**'i-lang'**, [**'lang'**],);  
 *// tmpObjStore.createIndex('i-lang-word', ['lang', 'word'], {  
 // unique: true  
 // });* tmpObjStore.createIndex(**'i-lang-isStudy'**, [**'lang'**, **'isStudy'**], {  
 **unique**: **false** });  
 *//-------------------------------------------------------------  
 // goto next* **case** 1 :  
 *//-------------------------------------------------------------  
 // Таблица "Переводы слов"* tmpObjStore = db.createObjectStore(**'WordsTranslate'**, {  
 **keyPath**: [**'langFrom'**, **'langTo'**, **'word'**],  
 **autoIncrement**: **false** });  
 *// Поиск всех слов по языку* tmpObjStore.createIndex(**'i-langFrom'**, [**'langFrom'**],);  
 *// Поиск всех переводов с основного языка на дополнительный* tmpObjStore.createIndex(**'i-langFrom-langTo'**, [**'langFrom'**, **'langTo'**]);  
 *// Поиск перевода для конкретного слова* tmpObjStore.createIndex(**'i-langFrom-langTo-word'**, [**'langFrom'**, **'langTo'**, **'word'**], {  
 **unique**: **true** });  
 *//-------------------------------------------------------------  
 // goto next* **default**:  
 **break**;  
 }

* “Books” – таблица для хранения книг, загруженных пользователем.  
  *// app/js/App/Idb/Books.js****App***.**Idb**.**Books** = {  
   */\*\*  
   \* Структура, описывающая хранимую книгу  
   \*/* **Struct**: {  
   */\*\*  
   \* Версия структуры на момент записи книги  
   \** ***@type*** *{Number}  
   \*/* **version**: 1,  
   */\*\*  
   \* Язык книги (Напр. "ENG" или "RUS")  
   \** ***@type*** *{String}  
   \*/* **lang**: **undefined**,  
   */\*\*  
   \* Заглавие книги на языке книги.  
   \*/* **title**: **undefined**,  
   */\*\*  
   \* BASE64-кодированное изображение титульной страницы.  
   \** ***@type*** *{String}  
   \*/* **image**: **undefined**,  
   */\*\*  
   \* MD5-хеш контента книги  
   \** ***@type*** *{String}  
   \*/* **hash**: **undefined**,  
   */\*\*  
   \* Контент FB2-книги  
   \** ***@type*** *{String}  
   \*/* **content**: **undefined** },

Первичный ключ: **keyPath**: **'hash'**

Индексы:  
«i-lang» - индекс по поля языку, на котором написана кгига. Позволяет выбирать только те книги, которые соответствуют изучаемому пользователем языку.

«i-title» - индекс по заглавию книги. Позволяет осуществлять поиск по заглавию.

* “KeyVal” – хеш-таблица общего назначения. Предназначена для хранения данных, которые по своей природе являются элементами «хеш-массива».  
  *// app/js/App/Idb/KeyVal/LastSession.js****App***.**Idb**.**KeyVal** = {  
   */\*\*  
   \* Ключи, которыми помечаются записи в таблице.  
   \*/* **KEYS** :{  
   */\*\*  
   \* Элемент хранит информацию о последней сессии пользователя  
   \*/* **LAST\_SESSION** : **'KEY\_LAST\_SESSION'** },

Первичным ключем является стандартный строковый ключ таблицы.

***App***.**Idb**.**KeyVal.KEYS** – перечисление допустимых ключей.

* + “LastSession” – единственная запись следующей структуры  
    *// app/js/App/Idb/KeyVal/LastSession.js****App***.**Idb**.**KeyVal**.**LastSession** = {  
     */\*\*  
     \* Структура, описывающая последнюю сессию пользователя.  
     \*/* **Struct**: {  
     */\*\*  
     \* Версия структуры на момент записи информации.  
     \** ***@type*** *{Number}  
     \*/* **version**: 1,  
     */\*\*  
     \* Язык пользовательского интерфейса (Напр. "ENG" или "RUS")  
     \** ***@type*** *{String}  
     \*/* **langGui**: **undefined**,  
     */\*\*  
     \* Изучаемый язык (Напр. "ENG" или "RUS")  
     \** ***@type*** *{String}  
     \*/* **langStudy**: **undefined**,  
     */\*\*  
     \* Идентификатор читаемой книги  
     \** ***@type*** *{String}  
     \*/* **bookHash**: **undefined**,  
     */\*\*  
     \* Позиция читаемой книги, на которой остановился пользователь  
     \** ***@type*** *{Number}  
     \*/* **bookPosition**: **undefined** },
* “WordsStudy” – таблица для хранения изучаемых и изученных слов.  
  *// app/js/App/Idb/WordsStudy.js  
  /\*\*  
   \*  
   \* Объект, предоставляющий интерфейс к таблице "Изучаемые слова".  
   \** ***@type*** *{Object}}  
   \*/****App***.**Idb**.**WordsStudy** = {  
   */\*\*  
   \* Структура, описывающая изучаемое слово.  
   \*/* **Struct**: {  
   */\*\*  
   \* Версия структуры на момент записи информации.  
   \** ***@type*** *{Number}  
   \*/* **version**: 1,  
   */\*\*  
   \* Язык слова (Напр. "ENG" или "RUS")  
   \** ***@type*** *{String}  
   \*/* **lang**: **undefined**,  
   */\*\*  
   \* Нормализованный текст слова (Напр. "dog", "яблоко")  
   \** ***@type*** *{String}  
   \*/* **word**: **undefined**,  
   */\*\*  
   \* Флаг: Слово для изучения? 0 - нет, 1 - Да.  
   \* 0 - слово уже изучено.  
   \* 1 - слово не изучено - прогнать через процесс изучения.  
   \** ***@type*** *{Number}  
   \*/* **isStudy**: **undefined**,  
   */\*\*  
   \* Флаг: Завершено ли изучение слова с родного языка на изучаемый?  
   \* 0 - Нет, 1 - Да  
   \** ***@type*** *{Number}  
   \*/* **isFinishedLang1ToLang2**: **undefined**,  
   */\*\*  
   \* Флаг: Завершено ли изучение слова с изучаемого языка на основной  
   \* 0 - Нет, 1 - Да  
   \** ***@type*** *{Number}  
   \*/* **isFinishedLang2ToLang1**: **undefined** }

Первичный ключ: композитный **keyPath**: [**'lang'**, **'word'**]

Индексы:

“i-lang” – позволяет искать слова по языку

“i-lang-isStudy” – позволяет искать изученные, либо неизученные слова для конкретного языка.

* “WordsTranslate” – таблица закешированных переводов слов.

*// app/js/App/Idb/WordsTranslate.js  
/\*\*  
 \* Объект, предоставляющий интерфейс к таблице "Перевод слов".  
 \** ***@type*** *{Object}}  
 \*/****App***.**Idb**.**WordsTranslate** = {  
 */\*\*  
 \* Структура, описывающая перевод слова.  
 \*/* **Struct**: {  
 */\*\*  
 \* Версия структуры на момент записи информации.  
 \** ***@type*** *{Number}  
 \*/* **version**: 1,  
 */\*\*  
 \* Для какого языка перевод слова (Напр. "ENG" или "RUS")  
 \** ***@type*** *{String}  
 \*/* **langFrom**: **undefined**,  
 */\*\*  
 \* На какой язык перевод слова (Напр. "ENG" или "RUS")  
 \** ***@type*** *{String}  
 \*/* **langTo**: **undefined**,  
 */\*\*  
 \* Нормализованный текст слова (Напр. "dog" или "яблоко")  
 \** ***@type*** *{String}  
 \*/* **word**: **undefined**,  
 */\*\*  
 \* Нормализованный перевод слова (Напр. "собака" или "apple")  
 \** ***@type*** *{String}  
 \*/* **translate**: **undefined**,  
 */\*\*  
 \* Частота употребления слова.  
 \* Дробное от 0 до 1. 1 = 100% означает, что слово имеет единствее значение.  
 \** ***@type*** *{Number}  
 \*/* **score** : **undefined**,  
 */\*\*  
 \* Время последней записи слова в таблицу (формат timestamp).  
 \** ***@type*** *{Number}  
 \*/* **insertAt**: 0  
 },

Первичный ключ: композитный **keyPath**: [**'langFrom'**, **'langTo'**, **'word'**]

Индексы:  
“i-langFrom” – Поиск всех слов по языку  
“i-langFrom-langTo” – Поиск всех переводов с основного языка на дополнительный

“i-langFrom-langTo-word” – Поиск перевода для конкретного слова на конкретном языке.

4.3. Конфигурация приложения.

Единственный файл конфигурации, влияющий на бизнес-логику приложения, это «app-env.json».

*// prototype\public\resources\app-env.json*{

**"defaults"**: {  
 **"langGui"**: **"RUS"**,  
 **"langStudy"**: **"ENG"** },  
 **"languages"**: {  
 **"RUS"**: **"Русский"**,  
 **"ENG"**: **"English"**,  
 **"UKR"**: **"Українська"** },  
 **"fb2"**: {  
 **"languages"**: {  
 **"ru"**: **"RUS"**,  
 **"en"**: **"ENG"**,  
 **"ua"**: **"UKR"** }  
 },  
 **"google"** : {  
 **"languages"** : {  
 **"RUS"** : **"ru"**,  
 **"ENG"** : **"en"**,  
 **"UKR"** : **"ua"** }  
 },  
 **"study"** : {  
 **"cntWordForStudy"** : 6,  
 **"cntCorrectForPass"** : 3  
 }  
}

**./defaults/** – языки, используемые по умолчанию (когда пользователь впервые попадает на сайт).

**./defaults/langGui** – язык пользовательского интерфейса (RUS - русский)

**./defaults/langStudy** – изучаемый пользователем язык (ENG - английский)

**./languages/** – перечень допустимых языков

**./fb2/languages/** – соответствие языков книг формата FB2 языкам сайта.

**./google/languages/** – соответствие языков Google языкам сайта.

**./study/** – раздел для конфигурации процесса изчения слов

**./study/cntWordForStudy/** – количество слов, выбираемых за один сеанс изучения (по умолчанию 6 слов, но пользователь может изменить значение).

**./study/cntCorrectForPass** – количество правильных ответов для того, чтобы засчитать слово как «заученное». По умолчанию нужно дать на 3 правильных ответа больше, чем неправильных.

4.4. Локализация приложения.

Локализационные файлы расположены в директории «prototype\public\resources\ localize» и представляют собой JSON-файлы. Название файла должно совпадать с кличем в секции «./languages/» конфигурационного файйла “app-env.json” (см п4.3).

Например: «ENG.json», «RUS.json» и т.д. Файлы в кодировке UTF-8.

Ключами файла является текст, используемый в программе (внутри HTML-тегов и в тексте скриптов JavaSxript, которые выводят пользователю информацию).

Например:

*// prototype\public\resources\localize\localize\ENG.json*

{  
 **"..."**: **"..."**,  
 **"Ok"**: **"Ok"**,  
 **"Читалка с изучением слов"**: **"Read book and words study"**,  
 **"Статистика слов"**: **"Statistic of words"**,  
 **"на изучении:"**: **"studing:"**,  
 **"изучено:"**: **"studied:"**,  
 **"Библиотека"**: **"Library"**,  
 **"Чтение"**: **"Reading"**,  
 **"Изучение слов"**: **"Words study"**,  
 **"Добавить книгу из файла"**: **"Add book from file"**,

*//...*

В момент запуска приложения определяется язык интерфейса для текущего пользователя. После этого синхронным AJAX запросом загружается необходимый файл локализации (синхронный потому что такой подход дает возможность локализовывать значения строковых свойств класса на этапе объявления класса, а не на этапе выполнения скрипта. Впрочем, если это не надо, то можно переделать метод загрузки локализационного файла на асинхронный).

Загрузка и кеширование локализации возлагается на класс «***App***.**Localize**» см.«\public\app\js\App\Localize.js». Единожды загруженный файл не загружается до перезапуска приложения и используется повторно при смене языка в разделе «Язык».

4.4.1. Локализация HTML

Для локализации HTML применен следующий трюк – каждый тег, содержащий текст для локализации, маркируется классом «class="need-translate"». В момент запуска приложения, либо в момент смены языка интерфейса, вызывается метод «App.localizeGui()». Этот метод находит нужные теги вызовом «document.querySelectorAll('.need-translate')», после чего подменяет innerHTML значением из локализационного файла.

В случае, если локализация выполняется впервые, ключ сохраняется в атрибут 'data-localize-key', откуда и используется в дальнейшем.

Такой подход дает возможность мгновенно перелокализовывать приложение без перезагрузки страницы.

4.4.2. Локализация в JavaScript

Обслущивается вызовом «App.localize», котрый обращается к «App.Localize.get» используя текущие настройки языка.

Например:

**throw new *App***.**Errors**.**User**.**PressCancel**(***App***.localize(**'Пользователь отменил изучение'**));

5. Работа с программой.

5.1. Первый запуск

При первом запуске приложения, система не найдет в локальной БД информации о пользователе и направит его в секцию установки языков вид которой представлен на Рис.3.

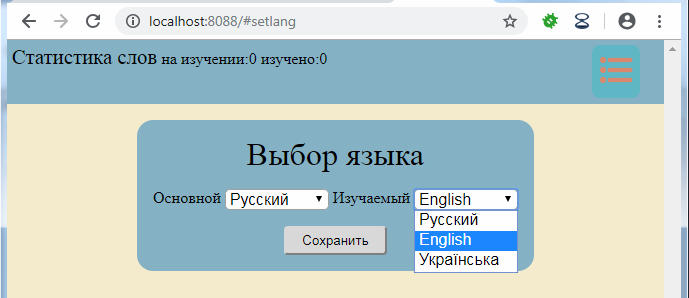


Рис.3 Первый запуск.

Тут мы видим форму выбора язвков, топ-панель компонентом «Статистика слов» и с кнопкой меню навигации.

Пользователь язык пользовательского интерфейса в поле «Основной», а также изучаемый язык в поле «Изучаемый».

После этого нажимает «Сохранить».

В этот момент:

* подгружается файл локализации
* перелокализовывается приложение (см. Рис4)
* выполняется навигация в секцию «Библиотека» «см. Рис5»

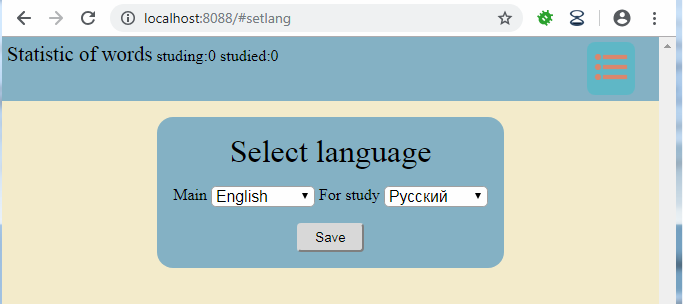


Рис.4 Секция «Установка языка» после перелокализации.

Форма выбора языков является комтонентом «setlang», который представлен скриптами:

\app\js\App\Component\Setlang.js

\app\css\component-setlang.css

5.2. Секция «Библиотека»

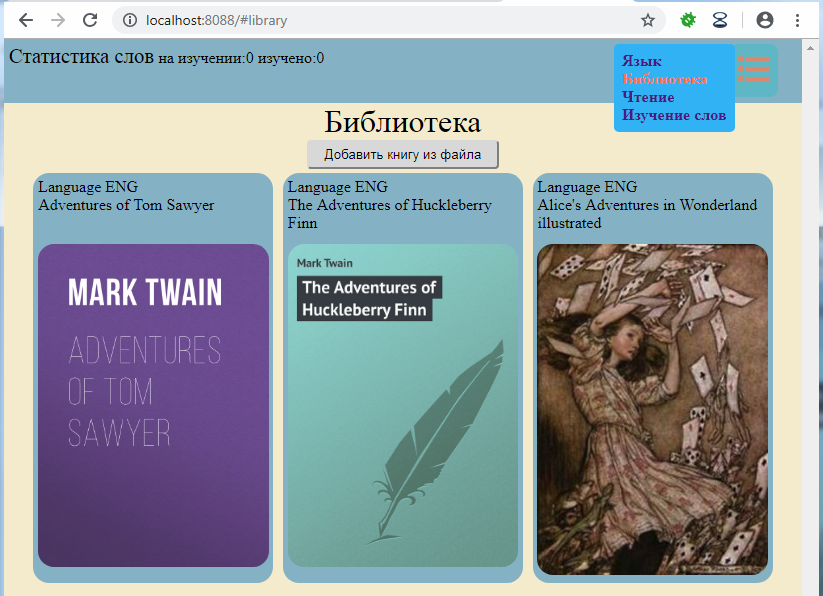


Рис.5 Секция «Библиотека».

Представляет собой компонент «library», построенный на CSS3 Flexbox.

Обслуживается скриптами:

* \app\css\component-library.css
* \app\js\App\Component\Library.js
* \app\js\App\Component\Library\BooksList.js
* \app\js\App\Component\Library\LoadFromFile.js

«BookList» и «LoadFromFile» являются субкомпонентами «Library».

При нажатии на кнопку «Добавить книгу из файла» пользователю будет предложено выбрать с диска книгу в формате FB2.

Со стороны HTML эта операция выполняется при помощи скрытого поля с типом “file”. Стандартное диалоговое окно на Рис.6

<**div class="library-loadfromfile"**>  
 <**div class="library-loadfromfile-wrapper"**>  
 <**button class="library-loadfromfile-button need-translate"**>Добавить книгу из файла</**button**>  
 <**input class="library-loadfromfile-input hidden" name="inpFile" type="file"**>  
 </**div**>  
</**div**>

Со стороны JavaScript чтение контента осуществляется при помощи вызовов:

***App***.**Component**.**Loadmask**.show(***App***.localize(**'Загрузка файла...'**));  
**Helper**.**Io**.loadTextFromInputFile(event.srcElement).then((text) => {  
 event.srcElement.**value** = **null**;  
 ***App***.**Component**.**Loadmask**.hide();  
 ***App***.bookToRead(text, **true**).then(() => {  
 ***App***.**Component**.**Nav**.go2section(**'read'**);  
 }).catch((e) => {  
 ***App***.**Component**.**Loadmask**.hide();  
 ***App***.**Component**.**WinMsg**.show({  
 **title**: ***App***.localize(**'Уведомление.'**),  
 **message**: (e **instanceof *App***.**Errors**.**User**) ? e.**message** : ***App***.localize(**'Ошибка при обработке книги'**)  
 });  
 });  
 ***App***.**Component**.**Library**.**isDisplayed** = **false**;  
}).catch((e) => {  
 ***App***.**Component**.**Loadmask**.hide();  
 **Helper**.**Log**.addDebug(e);  
});  
  
**Helper**.**Io**.loadTextFromInputFile = **function** (elFile) {  
 **return new *Promise***((resolve, reject) => {  
 **let** files = elFile.files;  
 **let** reader = **new *FileReader***();  
 reader.onload = (e) => {  
 resolve(e.target.result);  
 };  
 reader.onerror = () => {  
 reject();  
 };  
 reader.readAsText(files[0]);  
 });  
};

Тут мы видим сдедующее:

* Контент читается при помощи объекта FileReader
* Операция асинхронная и заворачивается в Promise
* На время загрузки и обработки файла вызывается модальная панель «Loadmask». См. Рис.7  
  Обслуживается скриптами:  
  \app\css\widget-loadmask.css  
  \app\js\App\Component\Loadmask.js
* В случае неудачи вызывается модальное диалоговое окно с уведомлением (Рис.8).  
   App.Component.WinMsg.show(...)  
  Обслуживается скриптами:  
  \app\css\win-msg.css  
  \app\js\App\Component\WinMsg.js
* В случае удачного чтения контента, его преобразование из FB2 во внутренний формат и сохранение в БД выполняется вызовом «App.bookToRead(...)». Детальнее об этом смотрите в п5.3
* В случае удачно парсинга управление передается компоненту-навигатору «App.Component.Nav.go2section('read')»

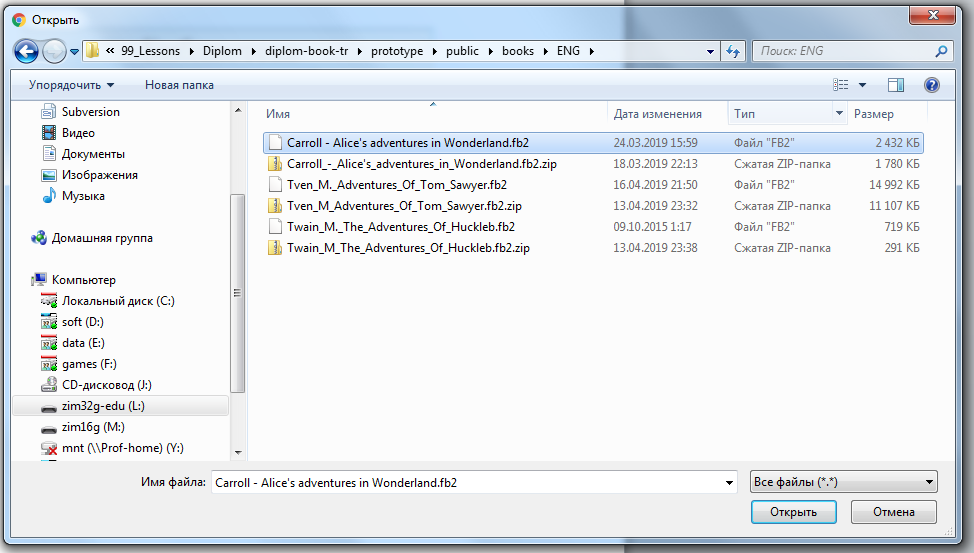
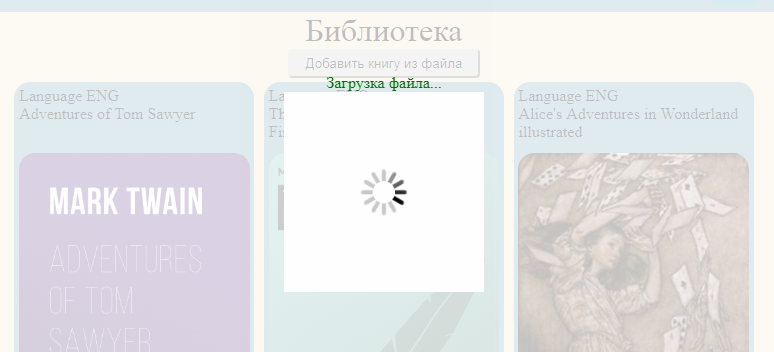
Рис.6 Секция «Библиотека», добавление книги.  


Рис.7 Компонент «Loadmask».

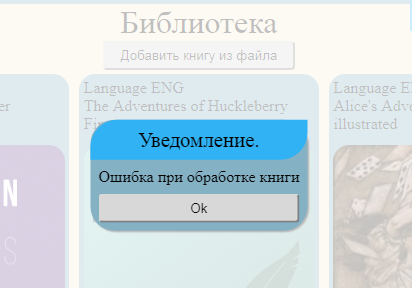


Рис.8 Компонент «WinMsg».

5.3. Секция «Чтение»

Сюда после клика по книге в секции «Библиотека», либо при нажатии кнопки «Наувигация/Чтение», либо при открытии браузера и наличии читаемой ранее книги. Общий вид представлен на Рис.9.

Обслуживается компонент следующими скриптами:

* \app\css\component-read.css
* \app\js\App\Component\Read.js

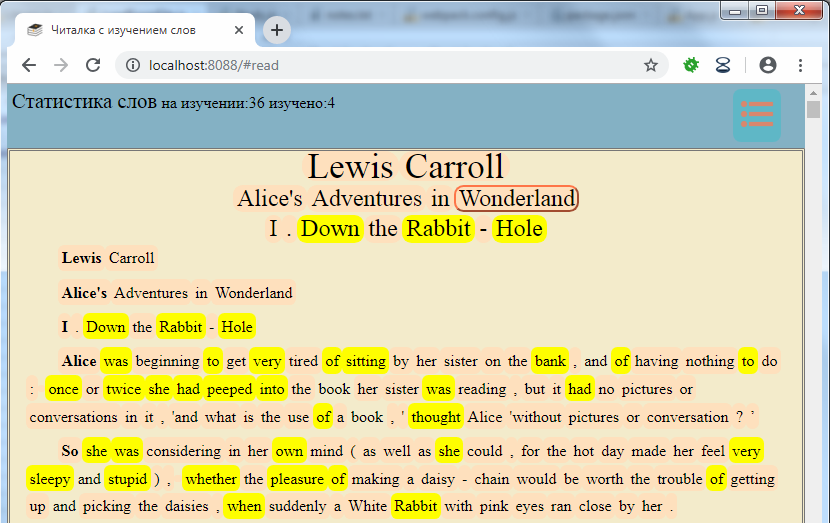


Рис.9 Вид компонента «Чтение».

Как мы видим, изучаемые слова помечаются желтым фоном, изученные имеют прозрачный фон, а неизвестные саетло-кирпичный. Тут может возникнуть вопрос, «А как же выделяются отдельные слова» ? Все просто – текст разбивается на заголовки, параграфы и отдельные слова (см. Рис.10). Каждое слово размещается в отдельном <span> с классами «read-word», «word-hash-...», «read-word-study| read-word-studied| read-word-unknown».

* «read-word» - класс общего назначения, он задает сформатирование слова в тексте. С его повощью можно выбрать все слова в тексте.
* «word-hash-...» - системный класс для поиска одинаковых слов. На место троеточия подставляется короткий хеш формата «hashCode» (см. **Helper**.**Hash**.hashCode). Он отлично подходит для данной задачи.
* «read-word-...» Маркировка слов по состоянию изучения
  1. «read-word-study» - слово изучается.
  2. read-word-studied» - слово изучено.
  3. read-word-unknown» - статус слова неопределен.

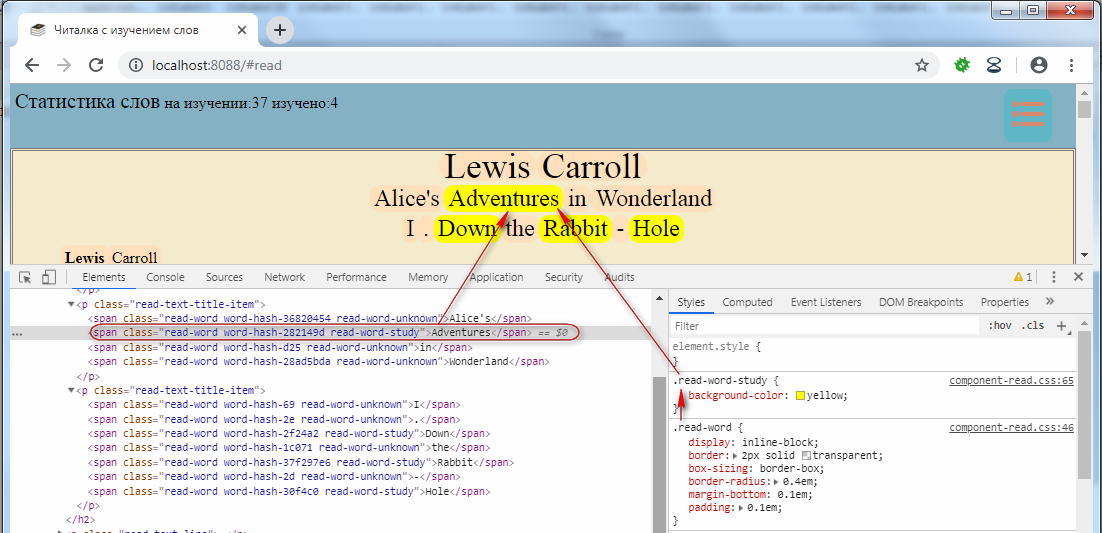


Рис.10 Внутреннее представление данных в компоненте «Чтение»

При наведении курсора на слово, оно выделяется рамкой и приобретает курсор вида “pointer”, побуждая пользователя выполнить клик. При клике на слово появляется модальное окно «Действия над словом» см. Рис.11

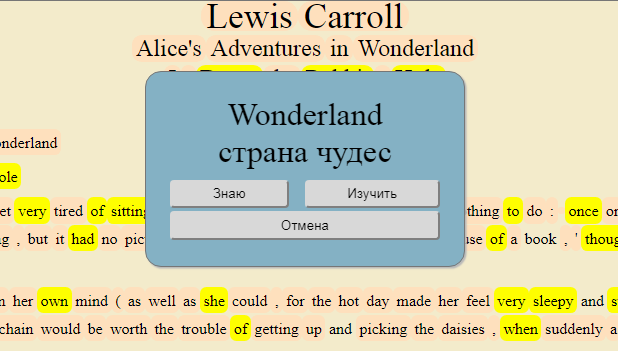


Рис.11 Вид компонента «Действия над словом».

В верхней части отображается выбранное пользователем слово, а в нижней перевод. Перевол получается по следующему алгоритму.

Сначала выполняется поиск в таблице «WordsTranslate» (вызов ***App***.**Idb**.**WordsTranslate**.get(...)). Если слово найдено, то возвращается перевод.

Если слово не найдено, то выполняется поиск перевода при помощи сервиса Google Translate (вызов **Helper**.**Google**.translate(...)).

Сервис доступен по URL следующего формата: <https://translate.googleapis.com/translate_a/single?client=gtx&sl=auto&tl=ru&dt=t&q=Wonderland>

Где в параметре «tl» следует задать язык, на котором хотим получить перевод, в параметре «q» следует указывать слово, которое переводим, а в параметре «sl» указвается язык оригинала (в данном случае автоматическое определение - auto). Полученный перевод имеет следующий формат:  
[[["страна чудес","Wonderland",null,null,1]],null,"en",null,null,null,0.87450981,null,[["en"],null,[0.87450981],["en"]]]

Вызовом метода **Helper**.**Google**.getTranslateConverted(...) получаем удобную для работы структуру и сохраняем в БД вызовом ***App***.**Idb**.**WordsTranslate**.put(...)

После этого перевод закеширован в БД и при дальнейшей работе обращения к Google Translate не производятся.

5.4. Секция «Изучение»

Для изучения выбраных слов пользователю следует нажать кнопку «Меню» , после чего нажать «Изучение слов» (см. Рис.12), будет выполнен переход в секцию «Изучение» (см. Рис.13).

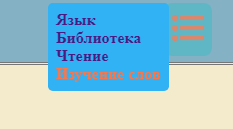


Рис.12 Навигация в секцию «Изучение».



Рис.13 Компонент «Изучение».

В нижней части можно увидеть субкомпонент «Грид», который отображает словарь пользователя и статус слов. Изученные слова отмечены серым кружком со знаком «-», изучаемые зеленым кружком со знаком «+». Грид наделен функционалом постраничного перехода, а также элементами для перехода на заданную страницу. Текущая страница и их общее количество выводятся в левом нижнем углу.

В верхней части можно увидеть область для инициализации процесса изучения. Поле «Кол-во слов для изучения» позволяет пользователю указать сколько слов нужно выбрать для изучения за один сеанс. Нажатие кнопки «Начать изучение» переводит в режим изучения (см. Рис.14).

Компонент «Изучение» обслуживается следующими скриптами:

* \app\css\component-study.css
* \app\css\component-study-grid.css
* \app\js\App\Component\Study.js
* \app\js\App\Component\Study\Grid.js

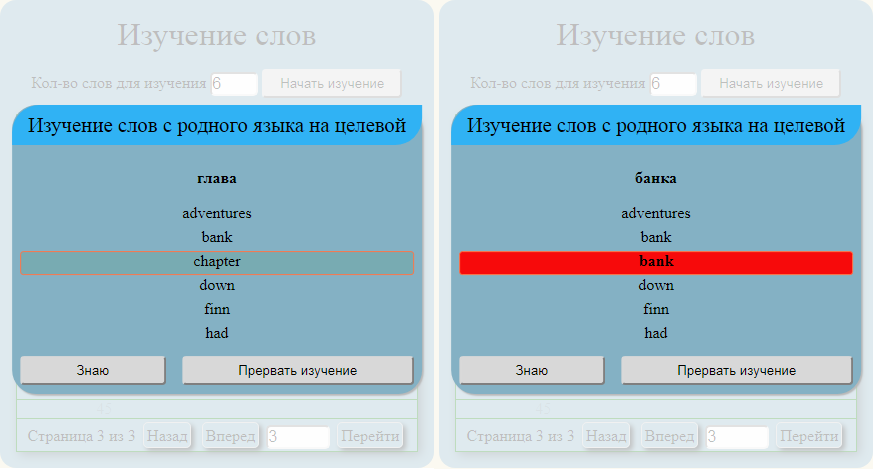


Рис.14 Компонент «Карточка изучения слов».

В верхней части окна выводится случайно выбраное слово для изучения. В центре варианты для выбора, в нижней части панель с кнопками. При наведении на слово в центральной части оно выделяется рамкой и курсором. Если пользователь нажал на слово, проверяется правильный ли вариант выбран. Если правильный, то на 1с включается анимация, плавно подсвечивающая слово зеленым фоном. Если же вариант неверен, то подсветка производится красным цветом, длится 1.5с, а вместо выбраного слова отображается правильный вариант. Правильный и неправильный ответы увеличивают внутренние счетчики на единицу. Для того, чтобы тест был пройден, пользователь должен для каждого слова дать на 3 правильных ответа больше, чем неправильных. Слово также считается пройденным, если пользователь нажал кнопку «Знаю». После изучения слов «С родного языка на целевой» языки меняются местами и пользователь изучает те же слова в режиме «Изучение слов с целевого языка на родной» (см. Рис.15).

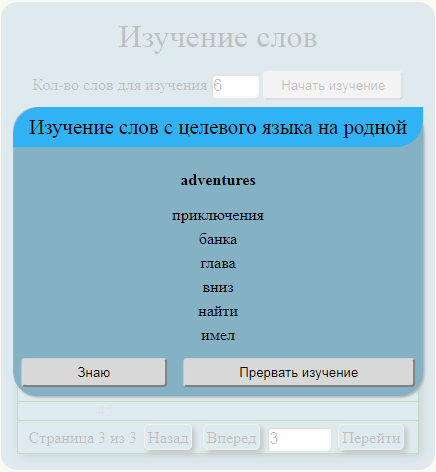


Рис.15 Реверсное изучение слов.

После изучения слов обновляется статистика в левом верхнем углу, перемаркировывается текст в мекции «Чтение» и обновляется грид со словарем пользователя.