

СОДЕРЖАНИЕ

1	СОЗДАНИЕ ИНТЕРФЕЙСА	2
2	СОЗДАНИЕ МАКЕТА БАЗЫ ДАННЫХ	7
3	ДОРАБОТКА МОДЕЛИ ДВИГАТЕЛЯ	9
4	НАПОЛНЕНИЕ БАЗЫ ДАННЫХ	14
5	СОЗДАНИЕ АНИМАЦИИ ПРОЦЕССОВ ОБСЛУЖИВАНИЯ.....	20
6	ДОБАВЛЕНИЕ ИНФОРМАЦИИ ОБ ИНСТРУМЕНТАХ.....	24
7	ДОБАВЛЕНИЕ ВОЗМОЖНОСТИ КОММЕНТИРОВАТЬ ПРОЦЕССЫ ОБСЛУЖИВАНИЯ	25
8	ДОБАВЛЕНИЕ АВТОРИЗАЦИИ.....	26
9	ДОБАВЛЕНИЕ АДАПТИВНОСТИ ПОД МОБИЛЬНЫЕ УСТРОЙСТВА	31
10	РАЗВЁРТЫВАНИЕ ПРИЛОЖЕНИЯ НА СЕРВЕРЕ.....	33
11	ПРОВЕРКА РАБОТЫ ПРИЛОЖЕНИЯ НА РАЗНЫХ УСТРОЙСТВАХ	35

Разработка состоит из следующих этапов:

1. Создание интерфейса.
2. Создание макета базы данных.
3. Доработка модели двигателя.
4. Наполнение базы данных.
5. Создание анимации процессов обслуживания.
6. Добавление информации об инструментах к процедурам обслуживания.
7. Добавление возможности комментировать процессы обслуживания.
8. Добавление авторизации.
9. Добавление адаптивности под мобильные устройства.
10. Развёртывание приложения на сервере.
11. Проверка работы приложения на разных устройствах.

1 СОЗДАНИЕ ИНТЕРФЕЙСА

Пользовательский интерфейс ИЭТР отображается через браузер. Для создания интерфейса использовался язык разметки HTML.

ЭСО состоит из блока для отображения 3д-модели, блока с информацией и блока навигации. При этом последние два блока являются частями одного блока `<div>`. Данные блоки позиционируются относительно друг друга с помощью сетки в CSS, известной как Grid layout. Параметры этой сетки заданы в файле `main.css`.

3д-модель отображается через тег `<div>`, который находится в левой части экрана и имеет атрибут `id` равный `“viewer”`. Когда пользователь открывает приложение, для него запускается клиентская часть, состоящая из html-кода страницы, css-стилей и клиентских скриптов на javascript. Скрипт `viewer.js` при запуске осуществляет следующие функции:

- 1) через jQuery берёт информацию о токене авторизации в платформе Forge;

- 2) создаёт объект Viewer, в который передаёт объект options, содержащий настройки и DOM-элемент, который является тем самым блоком для отображения 3д-модели;
- 3) запускает просмотр 3д-модели через метод viewer.start().

На рисунке 1 изображена функция, отображающая 3д-модель двигателя. Обратите внимание на метод Autodesk.Viewing.Document.load(), который вызывается в конце функции. Он загружает документ с заранее определённым идентификатором FORGE_MODEL_URN. Кроме того, в него передаются функции, которые срабатывают после загрузки документа или при ошибке его загрузки.

```
function loadModel() { //загружает модель без анимации
    $("#viewer").html(`</img>`); /*устанавливает изначальное содержимое блока,
    чтобы когда с документа обратно на 3д-модель переключаешься, логотип тоже был*/
    isModelLoaded = true;
    const htmlDiv = document.getElementById("viewer");
    const config = {
        extensions: ['Autodesk.Fusion360.Animation', 'Autodesk.NPR'], //тут подключаются расширения
        externals: { EventsEmitter: 'EventsEmmitter' }
    };

    viewer = new Autodesk.Viewing.GuiViewer3D(htmlDiv, config);
    const startedCode = viewer.start();
    if (startedCode > 0) {
        console.error("Failed to create a Viewer: WebGL not supported.");
        return;
    }

    console.log("Initialization complete, loading a model next...");
    viewer.addEventListener(Autodesk.Viewing.GEOMETRY_LOADED_EVENT, (e) => { //срабатывает после загрузки модели
        viewer.setLightPreset(8);
    });

    Autodesk.Viewing.Document.load(//загружает документ
        FORGE_MODEL_URN,
        onDocumentLoadSuccess,
        onDocumentLoadFailure
    );
}
```

Рисунок 1 – Функция loadModel(), отображающая 3д-модель

Блок с информацией об элементе ИЭТР отображается через тег <div> с атрибутом id равным “info”, который изначально пуст, но при загрузке скрипта main.js в него добавляется информация из базы данных, а именно общая информация о двигателе. Для этого в скрипте main.js вызывается функция showEngineDescription(), которая определена в скрипте database.js, так как она

осуществляет запрос в серверную часть, которая связывается с базой данных. Функция `showEngineDescription()` представлена на рисунке 2.

```
async function showEngineDescription() { //показывает описание двигателя
    $.get("http://localhost:3000/components", function (data) {
        $("#info").html(data[0].description);
    });
    if (animationLoaded) stopAnimation();
    if (!isModelLoaded) loadModel();
}
```

Рисунок 2 – Функция `showEngineDescription()`, отображающая описание двигателя

После GET-запроса через jQuery, который добавляет в блок для информации HTML-код с описанием двигателя, находятся две строки. Первая останавливает анимацию если она загружена. Это нужно, во-первых, чтобы пользователь не видел инструменты в разделах ИЭТР кроме процедур обслуживания, так как они там лишние, во-вторых, чтобы разделы «Состав двигателя» и «Информация о детали» корректно работали, так как каждой детали в Forge соответствует числовой номер, который присваивается автоматически при проходе системой сверху дерева модели вниз и при загрузке анимации он сбивается на 1, так как двигатель оборачивается в дереве в новый объект. Вторая строка загружает модель если она уже не загружена. Это нужно для того, чтобы когда пользователь открывает документ в окне просмотра, а потом переключает раздел ИЭТР, документ закрывался и отображалась 3д-модель.

Для остальных разделов ИЭТР созданы аналогичные функции в файле `database.js`, которые через запрос jQuery берут соответствующую информацию из ИБД, обрабатывают, если нужно, и устанавливают содержимое блока с информацией.

Все запросы к серверной части приложения осуществляются через протокол HTTP.

Блок навигации отображается через тег <div> с атрибутом id равным "sidebar". Изначально этот блок скрыт и отображается только при нажатии на кнопку «Содержание». Нажатие на эту кнопку вызывает функцию openNav(), которая устанавливает ширину блока навигации в 350 пикселей. Закрывается блок навигации нажатием на крестик, который устанавливает ширину в 0 пикселей. Ширина устанавливается через jQuery, команда выглядит следующим образом: «document.getElementById("mySidenav").style.width = "350px"».

Изначально в блоке навигации не установлены названия процедур обслуживания и ремонта, они берутся из базы данных через функцию showProcedures() файла database.js при запуске клиентской части приложения. Данная функция представлена на рисунке 3.

```
async function showProcedures() { //показывает список процедур в меню
  $.get("http://localhost:3000/procedures", function (data) {
    let mn = ``;
    let rep = ``;
    for (let i = 0; i < data.length; i++) {
      if (data[i].type == "maintenance") mn += `<li><a href="javascript:
      else if (data[i].type == "repair") rep += `<li><a href="javascript:
    }
    $("#maintenance").html(mn);
    $("#repair").html(rep);
  });
}
```

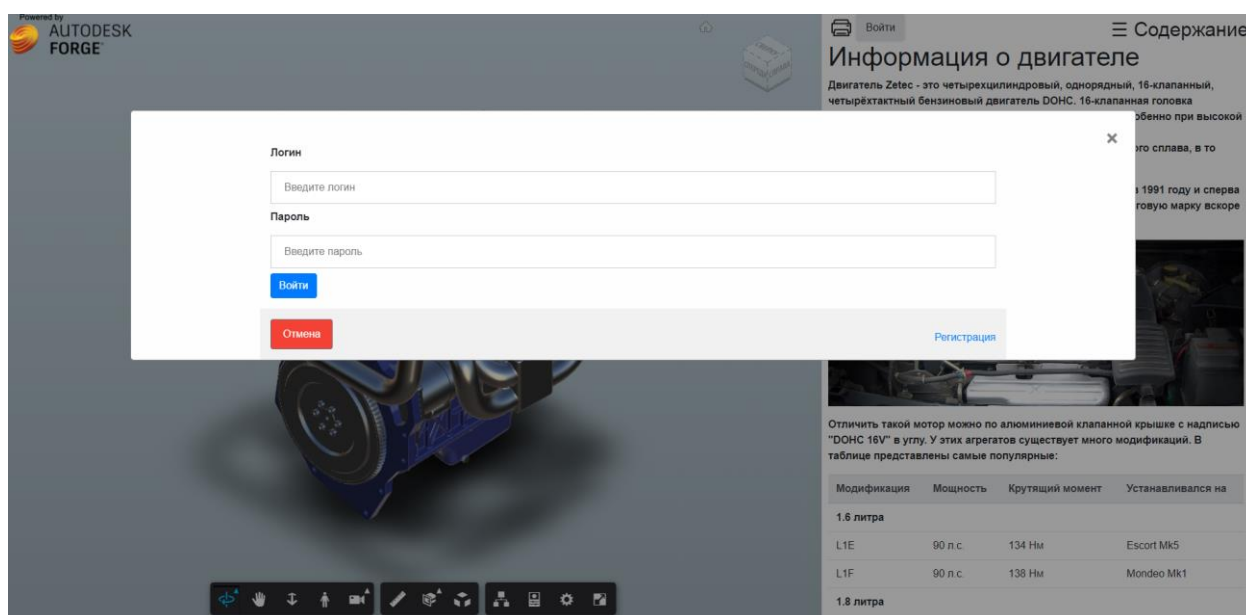
Рисунок 3 – Функция showProcedures(), отображающая список процедур обслуживания и ремонта

После того, как данные о процедурах получаются с серверной части в виде массива data, создаются две строковые переменные, в которые записывается HTML-код элементов списка, содержащих название процедуры и, при нажатии, отображающих страницу процедуры. Разбиение на две части происходит по типу процедуры, который записан в поле соответствующей таблицы ИБД.

Кроме того, интерфейс содержит кнопку печати, кнопку авторизации, форму авторизации и форму регистрации.

Кнопка печати вызывает функцию CallPrint() скрипта main.js, которая создаёт окно печати, в которое передаётся код блока с информацией, и открывает его.

Кнопка авторизации открывает форму авторизации, которая реализована как всплывающее окно. Данная форма представлена на рисунке 4.



Подробнее об авторизации рассказано в п. 8.

Форма регистрации аналогична форме авторизации, но в ней имеется также поле «Повторите пароль», а вместо кнопки «Войти» имеется кнопка «Регистрация».

HTML-код обеих форм описан в том же файле, что и остальная вёрстка приложения.

CSS-стили для интерфейса ИЭТР записаны в файлы:

1. “authorization.css”. Здесь хранятся стили для форм авторизации и регистрации.

2. “comments.css”. Здесь хранятся стили для отображения комментариев.

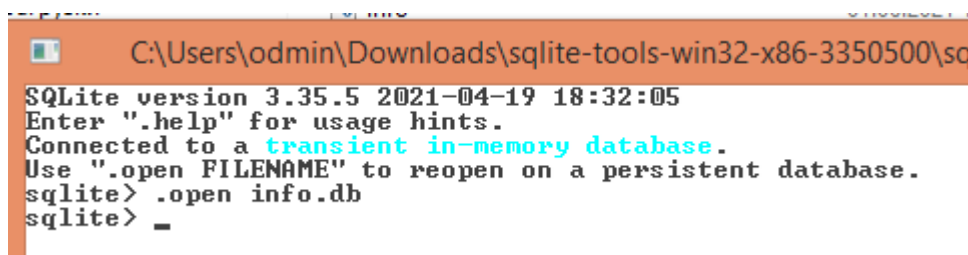
3. “main.css”. Здесь хранятся общие стили для области с текстом и области с 3д-моделью.

4. “navbar.css”. Здесь хранятся стили для панели навигации.

2 СОЗДАНИЕ МАКЕТА БАЗЫ ДАННЫХ

Для создания базы данных была выбрана СУБД “SQLite” версии 3 как наиболее простая в использовании. Кроме того, в этой СУБД нет необходимости настраивать сервер, у неё полностью свободная лицензия, она поддерживает кроссплатформенность и обладает высокой скоростью.

Для начала была скачана утилита Sqlite для ПК с официального сайта, в её консоли была введена команда для создания файла базы данных: «.open info.db». На рисунке 5 изображена эта команда.



```
C:\Users\odmin\Downloads\sqlite-tools-win32-x86-3350500\sq
SQLite version 3.35.5 2021-04-19 18:32:05
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open info.db
sqlite> _
```

Рисунок 5 – Создание файла базы данных

Затем, файл был открыт в программе «DB Browser for SQLite», которая также была скачана с официального сайта её разработчика. В программе созданы таблицы с полями, как изображено на физической схеме базы данных в п. 2.2. Структура БД в программе представлена на рисунке 6. Таблица sqlite_sequence создаётся программой автоматически.

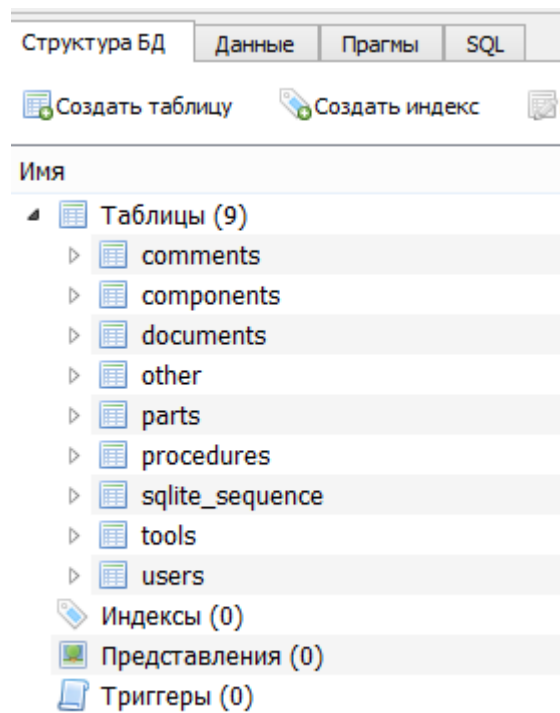


Рисунок 6 – Структура БД в программе «DB Browser for SQLite»

Для соединения базы данных с приложением в проект NodeJS была установлена и подключена в файле index.js библиотека “sqlite3”. После подключения базы данных в запускаящем файле были объявлены HTTP-запросы для взаимодействия с ней. Код подключения к БД и отправки из неё данных о компонентах двигателя в клиентскую часть показан на рисунке 7.


```

let sqlite3 = require('sqlite3').verbose();

//открытие базы данных
let db = new sqlite3.Database('info.db', sqlite3.OPEN_READWRITE, (err) => {
  if (err) {
    console.error(err.message);
  }
  console.log('Connected to the database.');
```

```

});

db.all("SELECT * FROM components", [], (err, rows) => { //получение компонентов
  if (err) {
    console.error(err.message);
  }
  app.get('/components', function (req, res) {
    res.send(rows);
  })
});

```

Рисунок 7 – Подключение к базе данных SQLite в модуле NodeJS

Аналогичным образом из базы данных получают данные о процедурах обслуживания, инструментах, расходных материалах, документах и прочем. Кроме того, серверная часть обрабатывает запросы клиента, связанные с учётной записью и с добавлением и удалением комментариев к процедурам обслуживания. Для этого клиент отправляет POST-запрос, содержащий информацию о том, какой комментарий нужно добавить или удалить, а при его принятии серверной частью вызывается соответствующий SQL-запрос к базе данных.

3 ДОРАБОТКА МОДЕЛИ ДВИГАТЕЛЯ

После изучения устройства двигателя и мануала по его обслуживанию выяснилось, что на модели, взятой с сайта GrabCAD, не хватает некоторых элементов, необходимых для демонстрации процедур обслуживания и собственно конструкции двигателя, а именно:

- 1) отсутствовала катушка зажигания;
- 2) отсутствовал масляный насос;

- 3) отсутствовали манжеты коленчатого вала и распределительных валов;
- 4) отсутствовала прокладка крышки головки блока цилиндров;
- 5) отсутствовали форсунки;
- 6) отсутствовали крышки шатунов;
- 7) отсутствовала сливная пробка;
- 8) конструкция ролика-натяжителя не соответствовала реальным образцам;
- 9) отсутствовали коренные подшипники распределительных валов и коленчатого вала;
- 10) отсутствовали болты крепления головки блока цилиндров, насоса системы охлаждения и масляного картера;
- 11) имелся не подключенный к источникам питания и не имеющий разъемов кабелепровод, ведущий к высоковольтным проводам зажигания и к корпусу выпускного коллектора;
- 12) отсутствовал масляный щуп;
- 13) масляный фильтр был не на том месте, где должен.

Кроме того, на скачанной модели было сцепление и коробка передач, а все детали не имели текстур. Коробка передач, как и сцепление, является лишней в руководстве по обслуживанию двигателя, к тому же она не имела внутренней детализации. Эти проблемы были исправлены в первую очередь: сцепление и коробка передач были удалены из модели, детали были окрашены в программе Fusion 360. На рисунке 8 показано сравнение модели до окрашивания деталей и доработки и после. Тут видно, как были раскрашены детали, добавлена катушка зажигания, убрано сцепление и добавлен масляный щуп.

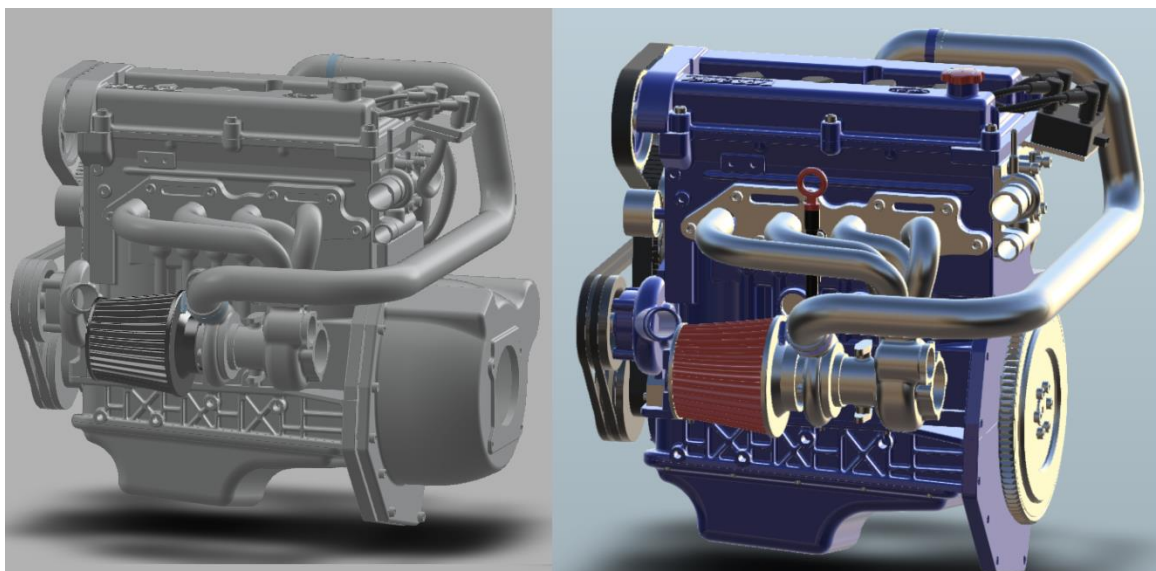


Рисунок 8 – Сравнение скачанной и доработанной моделей с передней стороны

На рисунке 9 изображено сравнение моделей: добавлен масляный насос, изменено положение масляного фильтра, добавлена сливная пробка.

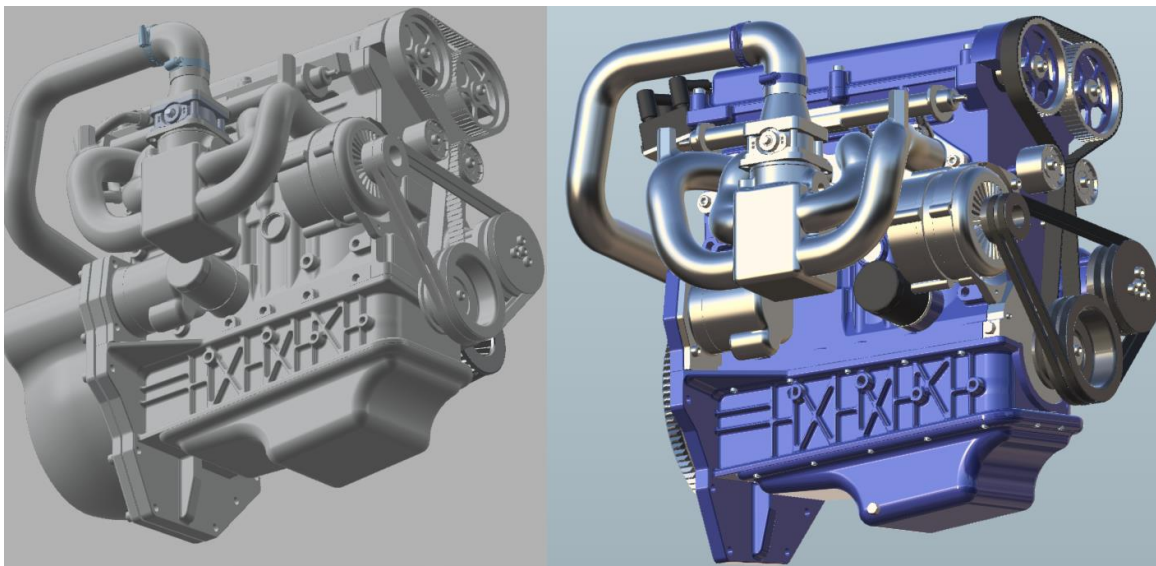


Рисунок 9 – Сравнение скачанной и доработанной моделей с задней стороны

На рисунке 10 показано внутреннее устройство модели двигателя: добавлена прокладка, сальники, подшипники, болты.

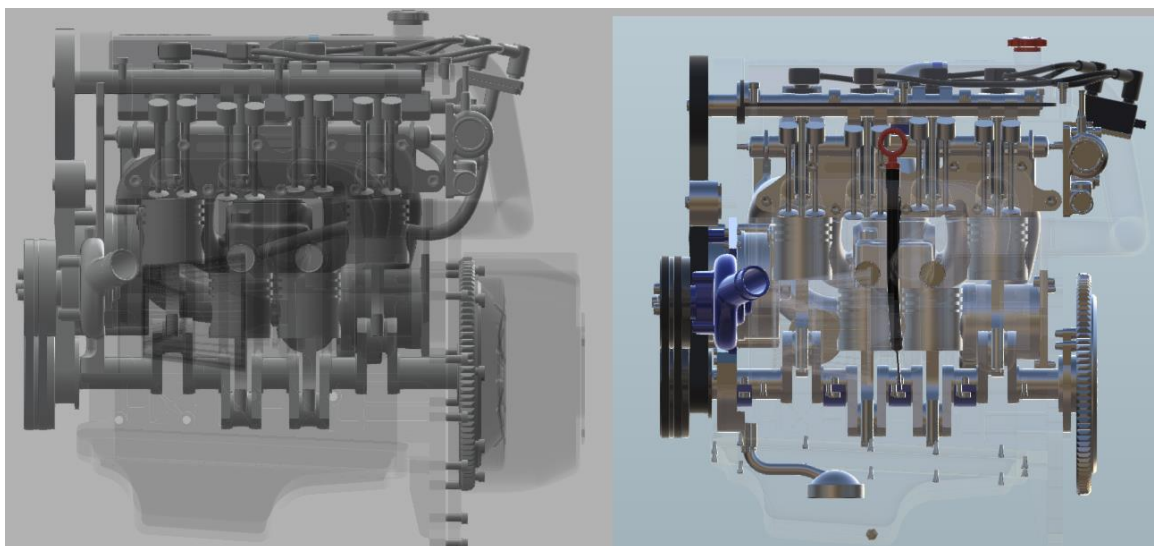


Рисунок 10 – Сравнение внутренних частей скачанной и доработанной моделей

Вдобавок, для показа анимации процедур обслуживания были использованы 3д-модели инструментов, а именно:

- 1) динамометрический ключ – скачан с GrabCAD;
- 2) ключ для масляного фильтра – скачан с GrabCAD и исправлен;
- 3) набор переходников для ключа – скачан с GrabCAD;
- 4) манометр для проверки компрессии – скачан с GrabCAD и исправлен;
- 5) набор щупов для проверки зазоров клапанов – смоделирован в Fusion 360;
- 6) шестигранный ключ – смоделирован в Fusion 360;
- 7) втулка для осадки сальников – смоделирована в Fusion 360;
- 8) стопор распределительных валов – смоделирован в Fusion 360;
- 9) бутылка с маслом – смоделирована в Fusion 360;
- 10) измеритель зазоров между электродами свеч зажигания – смоделирован в Fusion 360.

Модели инструментов хранятся в том же файле, что и модель двигателя, но скрыты и показываются только в нужный момент анимации. На рисунке 11 показаны инструменты для выполнения процедур обслуживания.



Рисунок 11 – Смоделированные инструменты для выполнения процедур обслуживания

Также, некоторые тела в модели были выведены в отдельные компоненты и группы, чтобы их можно было использовать в анимации. Для приведения модели двигателя в пригодный для отображения в ИЭТР вид были проведены следующие операции:

- 1) удалена коробка передач и сцепление;
- 2) смоделирована катушка зажигания;
- 3) смоделирован масляный насос с составными частями, при этом маслозаборник взят из другой модели двигателя;
- 4) смоделированы сальники валов и их корпуса;

- 5) смоделирована прокладка крышки блока цилиндров;
- 6) добавлены форсунки из другой модели двигателя;
- 7) смоделированы крышки шатунов;
- 8) смоделирована сливная пробка;
- 9) изменена конструкция ролика-натяжителя, чтобы можно было показывать анимацию ослабления и натяжения ремня ГРМ;
- 10) смоделированы крышки подшипников распределительных валов и коленчатого вала;
- 11) добавлены необходимые болты;
- 12) лишний кабелепровод удалён;
- 13) добавлен масляный щуп из другой модели двигателя;
- 14) масляный фильтр перемещён.

Все перечисленные процедуры по доработке модели проводились в соответствии со схемами, фотографиями и видеозаписями реальных двигателей Ford Zetec.

4 НАПОЛНЕНИЕ БАЗЫ ДАННЫХ

Наполнение базы данных производилось через программу “DB Browser for SQLite”. Сначала наполнялась таблица `components`. Иерархия компонентов была реализована в соответствии с деревом в 3д-модели, которое было приведено в более логичную форму в предыдущем пункте. Для того, чтобы компоненты можно было отображать иерархически, в таблице `components` были созданы поля `id` и `parent_id`. Поле `id` представляет собой уникальный идентификатор компонента, а `parent_id` – идентификатор родительского компонента, если он есть. Иерархия начинается с самого двигателя, поэтому родительский компонент есть у всех компонентов кроме первого – двигателя. В поле `node_ids` записывались номера тел (Body) в Forge Viewer. Эти номера зависят от положения тел 3д-модели в её дереве. Заполнение поля `node_ids` происходило следующим образом:

- 1) запускаясь приложение;
- 2) открывалась вкладка «Информация о детали»;
- 3) выделялись нужные детали, после чего в консоли интернет-обозревателя вводилась команда “viewer.getSelection()”.

Команда “viewer.getSelection()” отображает номера выделенных твёрдых тел и групп тел в виде массива чисел. Данный массив чисел потом можно использовать в командах “viewer.Select()” и “viewer.isolate()”, которые соответственно выделяют или изолируют указанные тела.

Данный способ заполнения позволяет легко прописывать в записях таблицы необходимые данные. Это очень важно, учитывая то, что в процессе доработки модели двигателя номера тел в Forge Viewer регулярно сбивались, так как их порядок и состав изменялся. Сначала предполагалось каждое твёрдое тело в модели записывать как отдельную запись, но таким образом получалось около пятисот записей при том, что название и описание у одинаковых элементов повторялись. Для ликвидации повторяющихся данных и сокращения числа записей была создана та структура хранения данных в таблице components, которая имеется в данный момент. Таблица components в программе просмотра базы данных представлена на рисунке 12.

Таблица: components

id	parent_id	node_ids	name	description
Фи...	Фильтр	Фильтр	Фильтр	Фильтр
1	1	NULL [1,2]	Ford Zetec	<h1>Информация о двигателе</h1>...
2	2	1 [3, 20]	Крышка головки блока цилиндров	<r>Деталь, закрывающая ...
3	3	2 [18, 19]	Прокладка крышки головки блока цилиндров	<r>Служит для изоляции внутренней части ...
4	4	2 [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]	Крышка масляной горловины	<r>Закрывает масляную горловину.</r>
5	5	1 [21]	Система зажигания	<r>Совокупность приборов и устройств, ...
6	6	5 [22, 23, 24, 25, 26, 27, 28, 29]	Резиновые колпачки высоковольтных проводов	<r>Подводят электричество к свечам ...
7	7	5 [30, 31]	Катушка зажигания	<r>Генерирует высоковольтный электрически...
8	8	5 [48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 6...	Свечи зажигания	<r>Устройство для воспламенения топлива...
9	9	1 [72]	Газораспределительный механизм	<r>Механизм, обеспечивающий впуск и выпус...
10	10	9 [73, 98]	Головка блока цилиндров	<r>Головка блока цилиндров (ГБЦ) ...
11	11	10 [78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 9...	Подшипники распределительных валов	<r>Подшипники скольжения, которые ...
12	12	10 [74, 75, 76, 77]	Сальники распределительных валов	<r>Манжета (сальник) – элемент для ...
13	13	9 [99, 100, 101, 102, 103, 104, 105, 106, 107, 108...	Клапаны	<r>Клапан — устройство, предназначенное д...
14	14	9 [131, 134, 135, 138, 139, 142, 143, 146, 147, 15...	Толкатели клапанов	<r>Толкатель — элемент ...
15	15	9 [195, 196, 197, 198, 199, 200, 201, 202, 203, 20...	Пружины клапанов	<r>Участвует в работе газораспределительно...
16	16	9 [227, 228, 239, 240]	Распределительные валы	<r>Распределительный вал (распредел) — в...
17	17	9 [229, 230, 231, 232, 233, 234]	Шкивы распределительных валов	<r>Передают вращение от ремня ...

Рисунок 12 – Таблица components ИБД ИЭТР

Для того, чтобы можно было отобразить список компонентов двигателя в иерархическом виде, недостаточно просто ввести поля `id` и `parent_id`. Для этого была создана функция “`sortComponents`” в файле `database.js`. Эта функция представлена на рисунке 13.

```
function sortComponents(data) { //сортирует массив компонентов в иерархическом виде
    let newdata = [];
    for (let i = 1; i < data.length; i++) {
        if (data[i].parent_id == 1) newdata.push(data[i]);
        else if (data[i].parent_id != 1) {
            let el = data.find((el) => el.id == data[i].parent_id);
            if (el.children == undefined) el.children = [];
            el.children.push(data[i]);
        }
    }
    return newdata;
}
```

Рисунок 13 – Функция для сортировки массива компонентов двигателя

Функция получает массив из компонентов двигателя и проходит по его элементам, при этом создаётся новый массив, в котором остаются компоненты верхнего уровня, а их дочерние компоненты помещаются в свойство “`children`” как массив. После этого функция возвращает отсортированный массив.

Для отображения компонентов используется функция “`showContents()`”. В ней есть переменная `contentstable`, которая содержит вёрстку таблицы на HTML и в неё в зависимость от уровня иерархии компонента записывается строка с соответствующим CSS-стилем. Код распределения компонентов на уровни представлен на рисунке 14.


```

data = sortComponents(data);
//добавление строк таблицы в соответствии с иерархией
for (let i = 0; i < data.length; i++) { //первый уровень иерархии
    contentstable += `<tr class="level1">
<td><a href="javascript:viewer.isolate(` + data[i].node_ids + `)">` + data[i].name + `</a></td>
<td>` + data[i].description + `</td>
</tr>`

    if (data[i].children != undefined)
        for (let j = 0; j < data[i].children.length; j++) { //второй уровень иерархии
            contentstable += `<tr class="level2">
<td><a href="javascript:viewer.isolate(` + data[i].children[j].node_ids + `)">` + data[i].children[j].name + `</a></td>
<td>` + data[i].children[j].description + `</td>
</tr>`;

            if (data[i].children[j].children != undefined) //третий уровень иерархии
                for (let k = 0; k < data[i].children[j].children.length; k++) {
                    contentstable += `<tr class="level3">
<td><a href="javascript:viewer.isolate(` + data[i].children[j].children[k].node_ids + `)">` + data[i].children[j].children[k].name + `</a></td>
<td>` + data[i].children[j].children[k].description + `</td>
</tr>`;
                }
        }
}

```

Рисунок 14 – Распределение компонентов на уровни для отображения в ИЭТР

Далее заполнялась таблица procedures. В ней указывалось название процедуры, её тип (ремонт или обслуживание), описание и всплывающие подсказки. Эти подсказки хранятся в поле annotations в виде массива в формате JSON. Это позволяет легко считывать массив в клиентской части. Заполненная таблица procedures показана на рисунке 15, а содержимое поля annotations – на рисунке 16.

proc_id	proc_name	type	annotations	description
Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1 Замена свечей зажигания	maintenance	[...]	<p></p>...
2	2 Замена масла и масляного фильтра	maintenance	[...]	<p></p>...
3	3 Замена сальников коленчатого вала	repair	[...]	<p></p>...
4	4 Замена сальников распределительных валов	repair	[...]	<p></p>...
5	5 Проверка и регулировка зазоров клапанов	maintenance	[...]	<p></p>...
6	6 Проверка компрессии в цилиндрах	repair	[...]	<p></p>...
7	7 Установка, осмотр и снятие масляного насоса	repair	[...]	<p></p>...

Рисунок 15 – Таблица procedures ИБД ИЭТР

```
[
  { "nodeid": 24, "start": 2, "end": 10, "text": "Снимите колпачок" },
  { "nodeid": 68, "start": 10, "end": 20, "text": "Выкрутите свечу" },
  { "nodeid": 21, "start": 20, "end": 30, "text": "Извлеките остальные  
свечи поочерёдно" },
  { "nodeid": 68, "start": 45, "end": 49, "text": "Осмотрите свечи" },
  { "nodeid": 56, "start": 50, "end": 59, "text": "Измерьте и  
отрегулируйте искровой промежуток" },
  { "nodeid": 21, "start": 60, "end": 70, "text": "Установите свечи и  
кабели в те же гнезда" }
]
```

Рисунок 16 – Поле annotations процедуры замены свечей зажигания

В этом поле для каждой всплывающей подсказки хранится свойство `nodeid`, указывающее на номер тела, к которому подсказка привязана, свойство `start`, указывающее на время появления подсказки в процентах от общего времени анимации, свойство `end`, указывающее на время исчезновения подсказки в процентах и свойство `text`, содержащее текст подсказки.

В приложении подсказки отображаются с помощью функции `animTick()`, которая срабатывает во время анимации 50 раз в секунду, обновляя положение подсказки. Функция представлена на рисунке 17.

```
function animTick() {
  if (animationLoaded) {
    let animExt = viewer.getExtension("Autodesk.Fusion360.Animation");

    let progress = Math.floor(animExt.getCurrentTime() / animExt.getDuration() * 100); //прогресс в процентах

    annotations.forEach(element => {
      let start = annotations[annotations.indexOf(element)].start; //начало аннотации в процентах
      let end = annotations[annotations.indexOf(element)].end; //конец аннотации в процентах
      if ((progress >= start) && (progress < end)) {
        displayAnnotation(annotations.indexOf(element));
      }
      if ((progress < start) || (progress >= end)) {
        hideAnnotation(annotations.indexOf(element));
      }
    });

    if (animExt.isPlaying()) {
      annotationUpdate();
    }
  }
}
```

Рисунок 17 – Функция для работы со всплывающими подсказками

Цикл проходит по массиву подсказок, который получается из базы данных и сравнивает их параметры start и end с текущим прогрессом анимации, который записывается в переменную progress.

Далее была заполнена таблица tools, в которой хранится информация об инструментах. Таблица представлена на рисунке 18.

id	name	procedure_ids	description	image
Фи...	Фильтр	Фильтр	Фильтр	Фильтр
1	Динамометрический ключ	[1,2,3,4,5,6,7]	Служит для затяжки резьбовых соединений с ...	
2	Ключ для масляного фильтра	[2]	Предназначен для снятия масляного фильтра.	
3	Набор щупов	[5]	Щупы предназначены для проверки и ...	
4	Манометр для проверки компрессии	[6]	Предназначен для определения давления в ...	
5	Стопор для распределительных валов	[7]	Представляет собой уголок толщиной 5 мм, ...	
6	Измеритель зазоров электродов свечей ...	[1]	Предназначен для измерения зазора между ...	
7	Шестигранный ключ	[4,7]	Предназначен для поворота ролика-...	
8	Приспособление для установки сальника	[3,4]	Представляет собой втулку с отверстием. ...	
9	Тонкий пластик	[3]	Используется для установки левой манжеты ...	

Рисунок 18 – Таблица tools ИБД ИЭТР

В поле procedure_ids записываются ID процедур обслуживания и ремонта, в которых данный инструмент употребляется.

Далее была заполнена таблица parts, в которой хранятся данные о расходных материалах для обслуживания двигателя. Таблица изображена на рисунке 19.

id	name	description	image	procedure_ids
Фи...	Фильтр	Фильтр	Фильтр	Фильтр
1	Масляный фильтр	В двигателях Ford Zetec используется масляны...		[2]
2	Левый сальник коленчатого вала	Сальник не позволяет машинному маслу ...		[3]
3	Сальники распределительных валов	Сальник не позволяет машинному маслу ...		[4]
4	Моторное масло	Всесезонное моторное масло, вязкость по ...		[2]
5	Герметик для двигателей	Высокотемпературный силиконовый герметик ...		[7]
6	Свеча зажигания	В 4-х цилиндровых двигателях Ford Zetec ...		[1]
7	Прокладка масляного поддона	Прокладка используется на двигателях выпуск...		[7]
8	Прокладка крышки головки блока цилиндров	Прокладка, используемая для уплотнения ...		[1,3,4,5,7]
9	Правый сальник коленчатого вала	Сальник не позволяет машинному маслу ...		[3]
11	Прокладка масляного насоса	Уплотняет соединение между корпусом ...		[7]
12	Прокладка маслозаборника	Уплотняет соединение между маслозаборнико...		[7]

Рисунок 19 – Таблица parts ИБД ИЭТР

Здесь в поле `procedure_ids` указаны ID необходимых процедур аналогично предыдущей таблице.

Далее были заполнены таблицы `users`, `documents` и `other` согласно схеме на рисунке 7. В таблице `users` создана запись администратора со значением поля `admin`, равным 1 и запись пользователя со значением поля `admin` равным 0.

Кроме того, ИЭТР содержит изображения, которые хранятся в папке “images”. Изображения взяты из трёх источников:

- 1) мануал по обслуживанию автомобилей Ford Mondeo;
- 2) открытая библиотека изображений без авторских прав Wikimedia Commons [34];
- 3) скриншоты деталей в Autodesk Fusion.

Скриншоты понадобилось делать для сальников, стопора распределительных валов и измерителя зазоров свечей зажигания, так как для них не нашлось изображений без авторских прав.

5 СОЗДАНИЕ АНИМАЦИИ ПРОЦЕССОВ ОБСЛУЖИВАНИЯ

Создание анимации к процессам обслуживания двигателя происходило в программе Autodesk Fusion 360. Для этого в программе открывался файл 3д-модели двигателя, после чего открывался режим анимации. Всего было создано 7 анимаций:

- 1) замена свечей зажигания;
- 2) замена масла и масляного фильтра;
- 3) проверка и регулировка зазоров клапанов;
- 4) замена сальников коленчатого вала;
- 5) замена сальников распределительных валов;
- 6) проверка компрессии в цилиндрах;
- 7) установка, осмотр и снятие масляного насоса.

Окно программы в режиме анимации показано на рисунке 20.

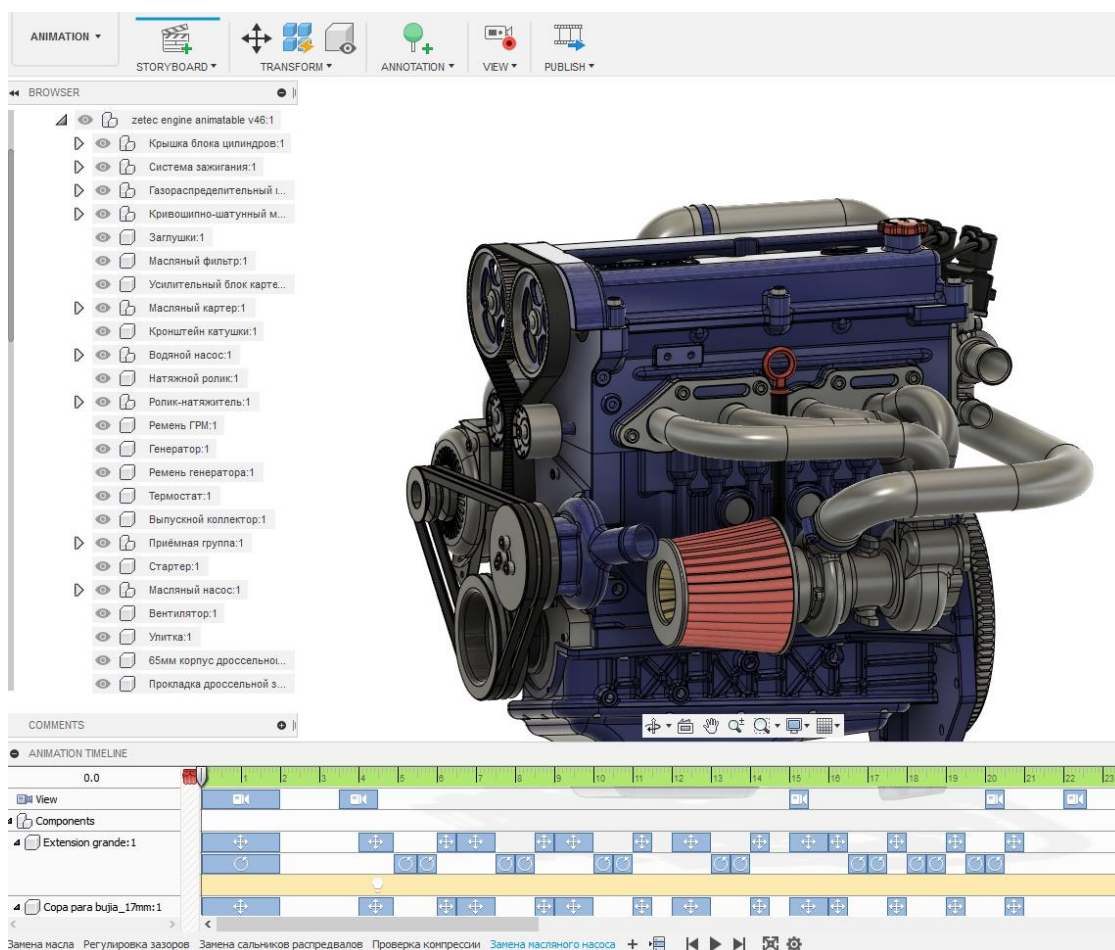


Рисунок 20 – Режим анимации в Autodesk Fusion 360

Сначала была создана анимация замены свечей зажигания. В ней по порядку продемонстрированы следующие операции:

- вынимаются колпачки свечей с высоковольтными проводами;
- выкручиваются свечи;
- измеряются зазоры электродов свечей с помощью приспособления;
- свечи зажигания и провода устанавливаются на место.

Потом была добавлена анимация замены масла. В ней показаны следующие операции:

- снимается крышка маслозаливной горловины;

- вынимается масляный щуп;
- вынимается сливная пробка для слива масла;
- пробка завинчивается, в маслозаливную горловину заливается масло;
- устанавливается масляный щуп и крышка горловины.

Поток машинного масла не показан, так как в Autodesk Fusion нельзя смоделировать поток жидкости в анимации.

Следом, создана анимация проверки и регулировки зазоров клапанов. Здесь показаны следующие операции:

- снимается крышка головки блока цилиндров;
- коленчатый вал устанавливается в положение верхней мёртвой точки для первого цилиндра;
- измеряются зазоры клапанов с помощью щупа;
- демонстрируется замена регулировочной шайбы толкателя клапана;
- устанавливается крышка головки блока цилиндров.

Затем создана анимация установки и снятия масляного насоса. Этот процесс необходим для последующей замены сальников коленчатого вала, но из-за своей трудоёмкости выделен в отдельную процедуру с отдельной анимацией. Здесь выполняются следующие операции:

- выкручиваются свечи зажигания;
- снимается крышка головки блока цилиндров;
- распределительные валы стопорятся;
- снимается ремень генератора;
- ослабляется натяжение ремня газораспределительного механизма (ГРМ);
- снимается шкив коленчатого вала;

- снимается ремень ГРМ;
- сливается масло из двигателя;
- вынимается масляный щуп;
- снимается масляный картер;
- снимается и разбирается масляный насос.

После этого происходит сборка в обратном порядке.

Далее была создана анимация процедуры замены сальников коленчатого вала. Она состоит из следующих операций:

- упрощённо снимается масляный насос, так как полное его снятие показано в другой анимации;
- заменяется правый сальник;
- снимается маховик;
- снимается корпус левого сальника;
- левый сальник заменяется, после чего устанавливается вместе с корпусом на двигатель;
- устанавливается маховик.

Далее идёт анимация замены сальников распределительных валов. В ней показаны следующие операции:

- снимается крышка головки блока цилиндров (ГБЦ);
- ослабляется ремень ГРМ;
- снимается шкив распределительного вала;
- снимается корпус подшипника и вынимается сальник;
- устанавливается корпус подшипника и новый сальник;
- устанавливается шкив распределительного вала;
- ремень ГРМ натягивается;
- устанавливается крышка ГБЦ.

В последней анимации, посвящённой проверке компрессии в цилиндрах, продемонстрированы следующие операции:

- выкручиваются свечи зажигания;
- компрессиометр устанавливается в отверстия свечей;
- вращается коленчатый вал.

Все анимации создавались в соответствии с руководством по ремонту автомобилей Ford Mondeo с двигателем Ford Zetec [25].

6 ДОБАВЛЕНИЕ ИНФОРМАЦИИ ОБ ИНСТРУМЕНТАХ

Информация об инструментах, используемых в обслуживании двигателя, отображается с помощью HTML-блока в начале страницы процедуры обслуживания. Блок содержит таблицу с инструментами. Отображение происходит через функцию “getProcedureTools” файла database.js. Часть функции, отображающая расходные материалы для процедуры, представлена на рисунке 21.

```
async function getProcedureTools(id) { //показывает инструменты и расходники на странице процедуры
  let info = `<h2>Вам понадобится:</h2>`;
  $.get("http://localhost:3000/parts", function (data) {
    let data1 = data.filter((val) => { return JSON.parse(val.procedure_ids).includes(id + 1) }); //фильтр расходников, связанных с процедурой
    if (data1.length != 0) {
      info += `<table class="table toolstable">
        <thead class="thead-light">
          <tr>
            <th scope="col">Расходник</th>
            <th scope="col">Фотография</th>
          </tr>
        </thead>
        <tbody>`;
      for (let i = 0; i < data1.length; i++) {
        info += `<tr>
          <td>+ data1[i].name + `</td>
          <td>+ data1[i].image + `</td>
        </tr>`;
      }
      info += `</tbody></table>`;
    }
  });
}
```

Рисунок 21 – Часть функции getProcedureTools, отображающая расходные материалы

Функция получает массив материалов с помощью GET-запроса с сервера через jQuery и фильтрует массив, чтобы получить только те

материалы, которые относятся к данной процедуре. Далее в строковую переменную `info`, содержащую HTML-вёрстку, добавляются строки таблицы, в которых имеются по две ячейки: название материала и его изображение.

Инструменты получаются, фильтруются и отображаются таким же образом. Содержимое переменной `info`, содержащей вёрстку двух таблиц, помещается в блок “tools” веб-страницы.

7 ДОБАВЛЕНИЕ ВОЗМОЖНОСТИ КОММЕНТИРОВАТЬ ПРОЦЕССЫ ОБСЛУЖИВАНИЯ

Для того, чтобы пользователи могли оставлять заметки, например, если им непонятен какой-то момент в выполнении процедуры, на страницы процедур был добавлен блок комментариев. Комментарии пользователей хранятся в ИБД в таблице `comments`. В блоке комментариев выводятся те, комментарии, которые добавлены под конкретно эту процедуру. У каждого комментария видно имя автора, время отправки комментария и сообщение. Кроме того, у обычных пользователей имеется кнопка «Ответить», а у администратора ещё и кнопка «Удалить». Добавлять комментарии могут только зарегистрированные пользователи.

Когда пользователь пишет сообщение и нажимает кнопку «Добавить комментарий», срабатывает функция “`addComment`”, которая отправляет на сервер POST-запрос, содержащий имя пользователя, текст сообщения, номер процедуры, к которой привязан комментарий и дату отправки. После этого сервер принимает запрос и добавляет комментарий в базу данных. Код обработки запроса сервером показан на рисунке 22.

```

app.post('/addComment', function (req, res) {
  db.serialize(function () {
    db.run('INSERT INTO comments (name, text, procedure_id, date) VALUES (?, ?, ?, ?)', [req.body.name,
      if (err) {
        return console.log(err.message);
      }
      console.log('Row was added to the table "comments": ` + JSON.stringify(req.body));
      res.send();
    });
  });
});

```

Рисунок 22 – Обработка сервером добавления комментария

После добавления комментария в базу данных клиент посылает GET-запрос и список комментариев обновляется.

8 ДОБАВЛЕНИЕ АВТОРИЗАЦИИ

Для того, чтобы разграничить права пользователей по добавлению комментариев и удалением оных, необходимо было ввести авторизацию.

В базе данных была создана таблица users, в которой хранятся данные пользователя: имя, пароль и права администратора. Записи о пользователях добавляются в базу данных через регистрацию. Форма регистрации представлена на рисунке 23.

Двигатель Zetec - это четырехцилиндровый, однорядный, 16-к...
 четырехтактный бензиновый двигатель DOHC. 16-клапанная го...
 рбен...
 его сл...
 199...
 гову...
 Кры...
 "DOHC 16V" в углу. У этих агрегатов существует много модифи...
 таблице представлены самые популярные:
 Модификация Мощность Крутящий момент Устан...

Рисунок 23 – Форма регистрации

После того, как пользователь ввёл логин и пароль в соответствующие поля и нажал кнопку «Регистрация», сервер получает POST-запрос, содержащий логин и пароль пользователя. Пароль зашифровывается с помощью библиотеки “bcryptjs”. Процесс регистрации пользователя в виде блок-схемы представлен на рисунке 24.

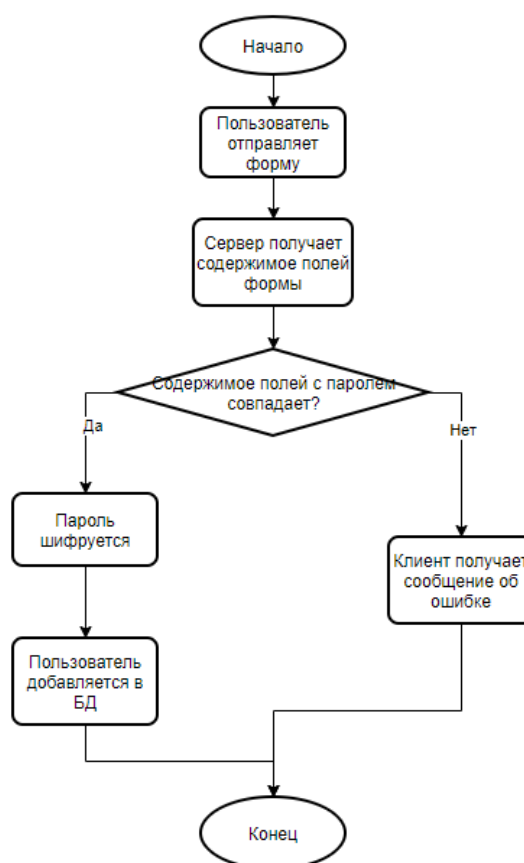


Рисунок 24 – Блок-схема регистрации пользователя

Зарегистрированному пользователю необходимо авторизоваться, чтобы оставить комментарий. Форма авторизации представлена на рисунке 14. Для авторизации пользователь заполняет поля «Логин» и «Пароль» и нажимает кнопку «Войти». При этом сервер получает POST-запрос, содержащий логин и пароль. Сервер сравнивает логин с логинами пользователей, которые есть в базе данных. Если такого пользователя нет, пользователю выводится ошибка. Если же есть, сервер сравнивает введенный пароль с зашифрованным паролем этого пользователя в БД. Если он совпадает, то авторизация завершается

успешно, если нет – пользователю выводится соответствующая ошибка. Процесс авторизации пользователя в виде блок-схемы представлен на рисунке 25.

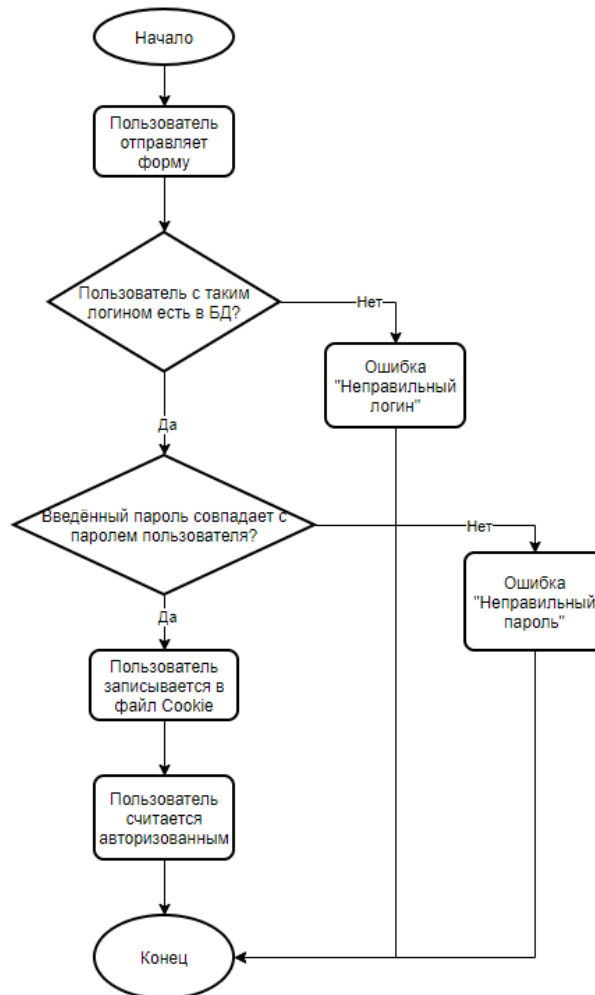


Рисунок 25 – Блок-схема авторизации пользователя

При успешной авторизации в Cookie-файл, хранящийся на устройстве пользователя, записывается имя пользователя и наличие у него прав администратора. Пользователь не может изменить эти значения. Данные сохраняются в файле Cookie в течение часа, после чего файл удаляется. Работа с Cookie производится с помощью библиотеки “client-sessions”. Запись данных сессии пользователя происходит так, как показано на рисунке 26. Данный процесс показан на примере сравнения введённого пользователем пароля с зашифрованным паролем пользователя в БД. Функция “isValidPassword”

осуществляет это сравнение. В неё передаётся пользователь из БД и введённый пароль.

```
if (isValidPassword(rows[0], req.body.password)) {  
    req.session_state.username = req.body.username;  
    req.session_state.admin = rows[0].admin;  
    res.send(req.body.username);  
}  
else {  
    console.error('Неправильный пароль');  
    res.sendStatus(400);  
}
```

Рисунок 26 – Запись данных в пользовательскую сессию

Предоставление пользователю возможности добавлять и удалять комментарии происходит в клиентской части через функцию “showComments”. Сначала функция получает текущего пользователя приложения. Если функция “getCurrentUser” возвращает пустое значение, на странице процедуры обслуживания в блоке для комментариев отображаются только сообщения. Если непустое, отображается также поле для добавления сообщения и кнопка отправки. Кроме того, если возвращаемое значение функции “getCurrentUser” говорит о том, что пользователь является администратором, у всех комментариев на странице появляется кнопка «Удалить». Подробнее этот процесс изображён на рисунке 27.

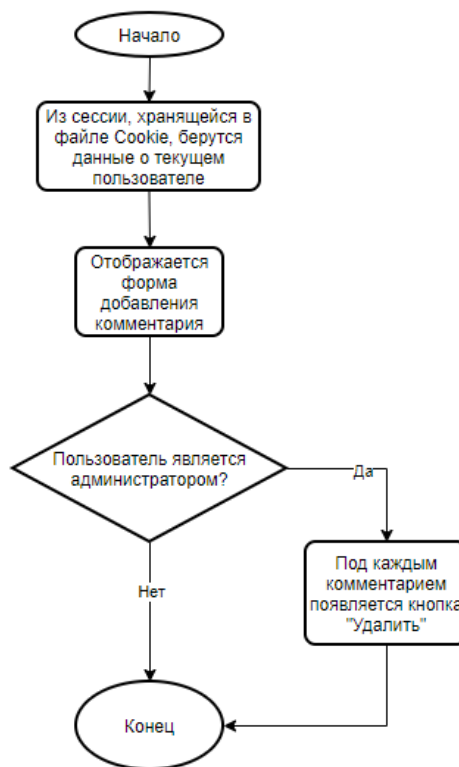


Рисунок 27 – Определение прав пользователя относительно комментариев

После предоставлении администратору возможности удалять комментарии было решено предоставить ему возможность просматривать и удалять учётные записи пользователей. Дополнительные элементы интерфейса было решено не добавлять, так как тогда приложение будет превращаться не в ИЭТР, а в социальную сеть. Администратор может просматривать список зарегистрированных пользователей с помощью команды “showUsers()”, которая вводится в консоль браузера. Удалять пользователя можно командой “removeUser(id)”, где id – идентификатор пользователя в базе данных. Подробнее процесс управления учётными записями пользователей представлен на рисунке 28.

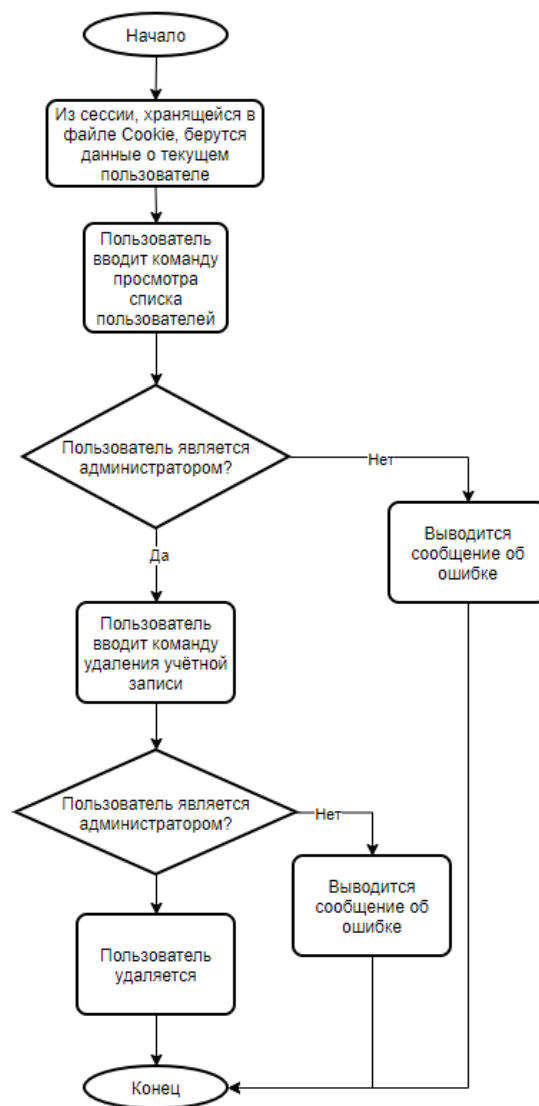


Рисунок 28 – Блок-схема управления учётными записями пользователей

9 ДОБАВЛЕНИЕ АДАПТИВНОСТИ ПОД МОБИЛЬНЫЕ УСТРОЙСТВА

Чтобы адаптировать дизайн приложения под все устройства, были созданы медиа-запросы на языке CSS. В файле `main.css` есть два медиа-запроса.

Первый запрос срабатывает при максимальной ширине экрана менее 1000 пикселей. Он помещает 3д-модель в верхнюю часть экрана, а блок информации – в нижнюю, чтобы на телефонах было удобнее читать текст, и сжимает кнопки интерфейса Forge, чтобы на телефонах они корректно отображались.

Второй запрос срабатывает при максимальной ширине экрана менее 1000 пикселей и альбомной ориентации экрана. Он нужен для того, чтобы на телефонах при альбомной ориентации интерфейс отображался как на компьютере, но с уменьшенным блоком 3д-модели и с увеличенным блоком информации.

В файле navbar.css есть такие же два запроса. Первый закрепляет кнопку открытия меню навигации в верхней части, чтобы она не закрывала часть текста на телефонах при портретной ориентации экрана. Второй наоборот, открепляет кнопку от верхней части, чтобы она всегда была в правом углу при альбомной ориентации на телефонах.

Пример отображения приложения на телефоне представлен на рисунках 29 и 30.

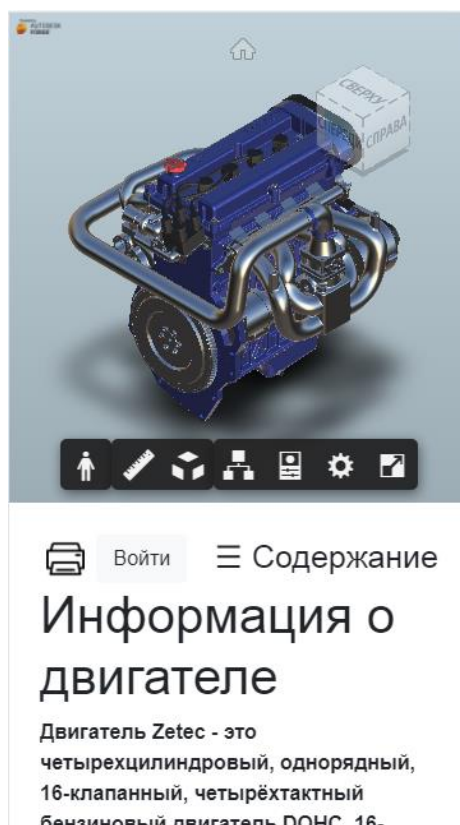


Рисунок 29 – Отображение на телефоне в портретной ориентации

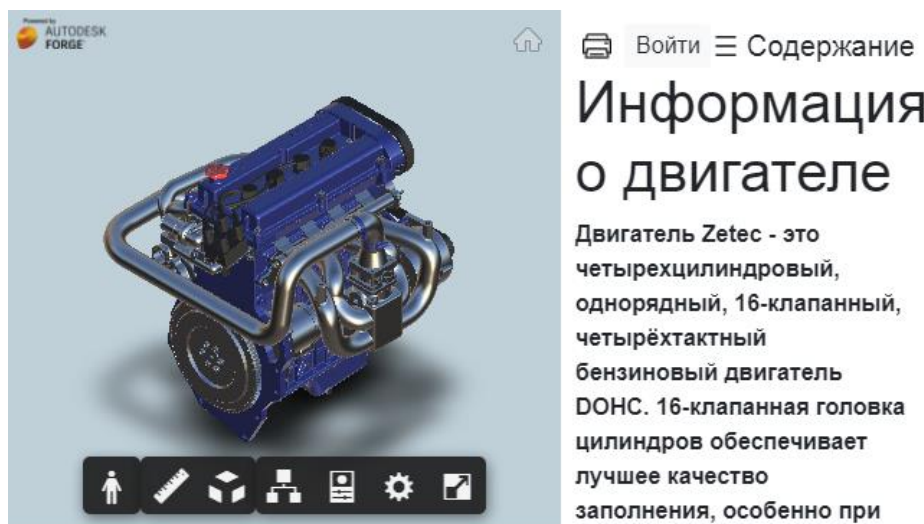


Рисунок 30 – Отображение на телефоне в альбомной ориентации

Далее было замечено, что на маленьких экранах таблицы в ИЭТР не помещаются полностью. Чтобы исправить это, таблицам была добавлена горизонтальная прокрутка. Для этого таблицы были помещены в HTML-блок, для которого установлен стиль “overflow-x: scroll”. Таким образом, если у пользователя не отображаются все колонки таблицы, он может прокрутить её вбок.

10 РАЗВЁРТЫВАНИЕ ПРИЛОЖЕНИЯ НА СЕРВЕРЕ

Для развёртывания приложения был использован сервер сайта, созданного центром САПР-разработки в Московском Политехе. На нём установлена ОС Windows Server. Доступ к серверу производился через удалённый рабочий стол, встроенный в ОС Windows. Работа осуществляется через HTTP-сервер Nginx. На рисунке 31 изображена часть конфигурационного файла Nginx, в которой настраивается домен и порт веб-приложения. В этом файле задаётся доменное имя приложения, по которому его можно открыть.

```
server {  
    listen 80;  
    server_name manual.mpu-cloud.ru;  
    location / {  
        proxy_set_header Host $host;  
        proxy_pass http://127.0.0.1:4015;  
        proxy_redirect off;  
    }  
}
```

Рисунок 31 – Настройки веб-сервера Nginx

Запуск приложения на сервере происходит через выполнение в командной строке Windows двух команд:

- переход в каталог, содержащий проект;
- запуск проекта Nodejs.

Эти команды записаны в файл .bat, который помещается в автозагрузку ОС.

Для открытия нужных портов на сервере был использован брандмауэр Windows. Настройки правила для открытия порта 4015 в брандмауэре Windows показаны на рисунке 32.

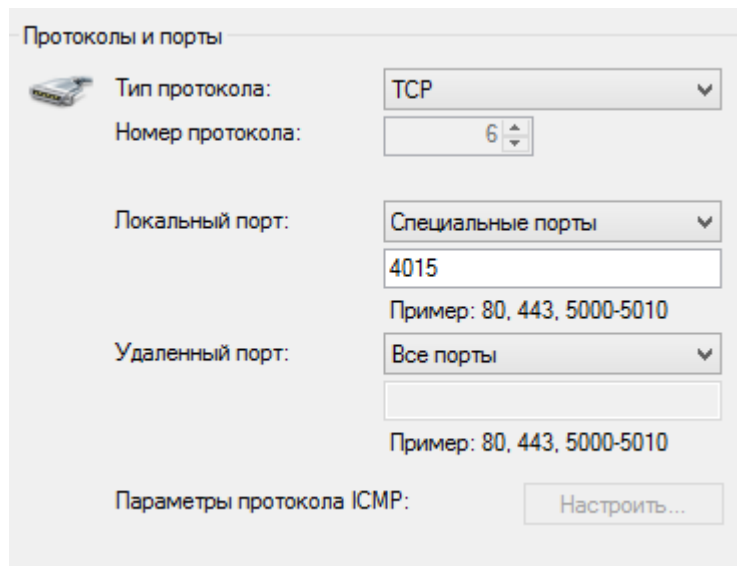


Рисунок 32 – Настройки правила в брандмауэре Windows

Также приложение было развёрнуто на облачной платформе Heroku. Для этого проект Heroku был подключен к нужному проекту на GitHub и произведено развёртывание.

11 ПРОВЕРКА РАБОТЫ ПРИЛОЖЕНИЯ НА РАЗНЫХ УСТРОЙСТВАХ

Приложение было протестировано на следующих устройствах:

1. Персональный компьютер с ОС Windows, браузером Google Chrome и 8 Гб оперативной памяти. В анимации процедуры «Установка, осмотр и снятие масляного насоса» в конце обнаружено мерцание деталей.
2. Смартфон Huawei Honor 5X с браузером Google Chrome и 2 Гб оперативной памяти. Постоянное мерцание деталей в анимациях, долгая загрузка модели.
3. Смартфон Realme 6 Pro с браузером Google Chrome и 8 Гб оперативной памяти. Периодическое мерцание деталей.
4. Планшет Asus MeMo Pad HD 7 с браузером Google Chrome и 1 Гб оперативной памяти. Модель не загружается.

Мерцание деталей возникало из-за включенной опции активного отображения в настройках Forge. Проблема была решена путём отключения этой настройки по умолчанию. Других проблем замечено не было. В результате приложение стало работать на устройствах следующим образом:

1. Персональный компьютер с ОС Windows, браузером Google Chrome и 8 Гб оперативной памяти. Работает стабильно.
2. Смартфон Huawei Honor 5X с браузером Google Chrome и 2 Гб оперативной памяти. Низкая скорость анимации.
3. Смартфон Realme 6 Pro с браузером Google Chrome и 8 Гб оперативной памяти. Работает стабильно.
4. Планшет Asus MeMo Pad HD 7 с браузером Google Chrome и 1 Гб оперативной памяти. Модель не загружается.
5. Смартфон Samsung Galaxy A51 с браузером Google Chrome и 4 Гб оперативной памяти. Работает стабильно примерно в 20 кадров в секунду.

Таким образом было установлено требование к объёму оперативной памяти на устройстве для запуска приложения: минимальный – 4 Гб, рекомендуемый – 8 Гб.