ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ МОЛДОВЫ

ФАКУЛЬТЕТ МАТЕМАТИКИ И ИНФОРМАТИКИ
ДЕПАРТАМЕНТ ИНФОРМАТИКИ

Ecaterina Cazac
IA 2304

Индивидуальная работа

по дисциплине „ Development of Server-Side Applications”

Руководитель: N.Nartea
(подпись)

Автор: Ecaterina Cazac
(подпись)

Кишинев, 2025

Теоретическая часть

- Введение в Express.js. Создание простого приложения "ToDo List"

Express.js - это минималистичный и гибкий фреймворк для Node.js, предназначенный для создания веб-приложений и API. Он упрощает работу с маршрутизацией, обработкой запросов, middleware и шаблонизаторами.

Одним из базовых подходов к организации кода в Express является архитектура MVC (Model-View-Controller), которая разделяет приложение на три слоя:

- Model: управляет данными и бизнес-логикой.
- View: отвечает за отображение данных пользователю.
- Controller: обрабатывает входящие запросы, вызывает методы модели и выбирает представление.

Такое разделение упрощает поддержку, масштабирование и тестирование приложения.

В рамках данной лабораторной работы рассматривается создание простого приложения «ToDo List», реализованного с использованием Express.js и архитектуры MVC.

- Понять базовую связку Express + MVC: контроллеры, роуты, представления.
- Научиться обрабатывать формы (GET/POST), передавать данные в шаблоны и делать redirect после успешной отправки.

Формулировка задачи

Необходимо разработать минимальное приложение ToDo List с возможностями:

- Просмотра списка задач.
- Создания новой задачи.
- Переключения статуса задачи (выполнена / не выполнена).
- Удаления задачи.

Приложение должно работать без базы данных, с хранением данных в памяти процесса.

Описание цели и основных этапов работы

Цель работы:

- Освоить принципы архитектуры MVC и базовую структуру приложений на Express.js.
- Научиться обрабатывать формы (GET / POST), передавать данные в шаблоны и выполнять redirect после успешной отправки формы.
- Реализовать простое серверное приложение с шаблонами и хранением данных в оперативной памяти.

Основные этапы выполнения:

1. Инициализация проекта - настройка окружения Node.js, установка зависимостей, создание структуры каталогов.
2. Настройка Express - подключение шаблонизатора EJS, middleware, статических файлов и маршрутов.
3. Реализация логики по MVC - создание моделей, контроллеров и представлений для работы со списком задач.
4. Обработка форм - реализация добавления, переключения статуса и удаления задач с помощью методов POST.
5. Тестирование работы приложения - проверка маршрутов, форм и отображения данных в браузере.

Практическая часть

В ходе выполнения лабораторной работы было создано приложение ToDo List, реализующее архитектурный подход MVC на платформе Express.js. Приложение предоставляет возможность пользователю управлять задачами - просматривать, добавлять, переключать статус (выполнена/не выполнена) и удалять.

Ключевые особенности реализации:

Хранение данных в памяти - задачи сохраняются в массиве (без использования базы данных). При перезапуске сервера данные сбрасываются, что соответствует целям учебной лабораторной работы.

Архитектура MVC - код разделён на логические слои:

- Model - описывает структуру данных (список задач) и операции с ними.
- Controller - обрабатывает запросы пользователя и управляет моделью.
- View - отвечает за отображение данных, используя шаблоны EJS.

Шаблонизатор EJS - для генерации HTML-страниц с динамическим контентом. Используются частичные шаблоны (partials/header.ejs, partials/footer.ejs) для единого макета.

Middleware Express - применяются morgan, express.urlencoded, express.static.

Redirect после POST - реализован паттерн Post/Redirect/Get для предотвращения повторной отправки формы при обновлении страницы.

Генерация уникальных ID - используется библиотека uuid для присвоения идентификаторов задачам.

Обработка ошибок 404 - реализована через отдельный контроллер errorController.js.

Приложение проверено на корректность работы всех маршрутов и форм:

GET / - список задач

GET /new - форма добавления

POST /new - создание задачи

POST /:id/toggle - изменение статуса

POST /:id/delete - удаление

GET /about - страница с информацией о приложении

Обработка несуществующих маршрутов - 404.ejs

Вывод

В результате выполнения лабораторной работы создано серверное приложение ToDo List, реализованное на Node.js с использованием фреймворка Express.js и архитектуры MVC. Были изучены и применены следующие навыки:

- настройка Express-приложения и организация структуры проекта;
- работа с шаблонизатором EJS и отображением динамических данных;
- обработка форм (GET/POST) и редирект после отправки;
- хранение данных в памяти и работа с массивами объектов;
- реализация маршрутов и контроллеров;
- подключение middleware для логирования и парсинга запросов.

Приложение успешно реализует все заявленные функции.

https://github.com/katerinnka/todo-app-express-mvc.git

Ответы на контрольные вопросы

1. Чем отличаются HTML-маршруты от REST API?
HTML-маршруты возвращают готовые HTML-страницы, формируемые с помощью шаблонов (res.render), и используются в веб-приложениях.
REST API-маршруты возвращают данные в формате JSON (res.json) и предназначены для взаимодействия с клиентскими приложениями (например, SPA или мобильными).

2. Что такое res.render и res.json? Когда использовать?

res.render(view, data) - рендерит HTML по шаблону и используется для отображения страниц пользователю.

res.json(data) - возвращает JSON-объект и применяется в API или AJAX-запросах.

3. Что такое middleware в Express и зачем используется express.urlencoded?
Middleware - это функции, обрабатывающие запросы между клиентом и конечным маршрутом. Они позволяют добавлять функциональность: логирование, аутентификацию, парсинг данных.
express.urlencoded({ extended: true }) используется для обработки данных, отправленных через HTML-формы методом POST, и позволяет получить их в req.body.

Список использованных источников

- https://nodejs.org/en
- https://expressjs.com/