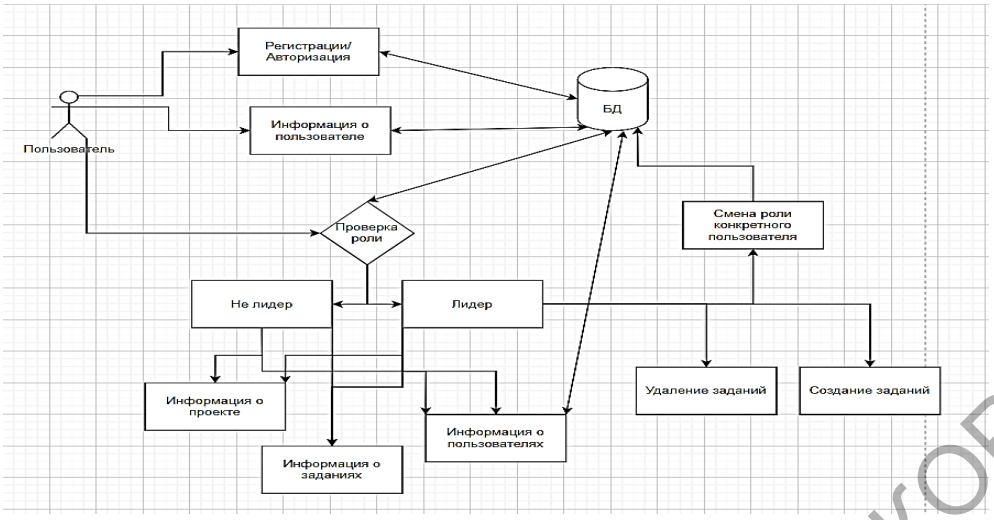

Рисунок 1 – Схема взаимодействия серверной и клиентской частей

Так как приложение разработано в AEM, то его может доработать человек, не обладающий знаниями в сфере разработки, с помощью добавления контента. Также программное обеспечение полностью адаптировано под мобильные устройства, что позволяет пользоваться им с любого устройства. Основные задачи, которые выполняет разработанное веб-приложение: 1) вход и регистрация пользователей; 2) форма создания задания с такими полями, как название, короткое описание, тип задания и пользователи, на которых назначается это задание; 3) отображение этих заданий, сортировка заданий; 4) управление ролями в проекте; 5) удаление заданий и несколько вспомогательных компонентов.

Описываемое приложение прошло апробацию в коллективной разработке небольшого веб-приложения, состоящего из трёх разработчиков. Были отмечены положительные моменты от его использования: удобный и понятный интерфейс, полная адаптация под мобильные платформы, идеально подходит для разработки чего-либо в небольшой группе людей. Были высказаны пожелания на улучшение проекта: добавить больше возможностей сортировки заданий, разработать систему историй статусов для заданий.

Результаты работы в виде готового рабочего приложения находятся в сети Интернет по адресу: https://github.com/dimaxdqwerty/project_management_app.

Литература

1 Документация AEM [Электронный ресурс]. – Режим доступа : https://experienceleague.adobe.com/docs/experience-manager-cloud-service.html. – Дата доступа : 28.04.2022.

2 Документация Sling [Электронный ресурс]. – Режим доступа : https://sling.apache.org. – Дата доступа : 28.04.2022.

УДК 004.415.53

А. С. Зайцев

РЕАЛИЗАЦИЯ ТЕСТИРОВАНИЯ РЕШЕНИЯ ЗАДАЧИ ВЕБ-ПРИЛОЖЕНИЕМ LEARN_JS ПО ИЗУЧЕНИЮ JAVASCRIPT

Статья посвящена описанию разработанного в кроссплатформенной среде WebStorm и документоориентированной базе данных MongoDB клиент-серверного

приложения Learn_JS для изучения синтаксиса языка программирования JavaScript и основ алгоритмизации путем решения задач. Приведена реализация автоматизации проверки выполненного практического задания обучающимся и результаты апробации приложения.

При большом интересе к онлайн-обучению очевидно, что освоить язык программирования без практики невозможно. Работа посвящена описанию разработанного клиент-серверного приложения Learn_JS, начало разработки которого описано в [1], для изучения синтаксиса языка программирования JavaScript [2] и основ алгоритмизации путем решения доступных в приложении задач. Для достижения цели необходимо было решить задачи: спроектировать базу данных; разработать методику автоматизации тестирования решения; разработать тесты; разработать необходимые функциональные части приложения (авторизация и регистрация пользователя, просмотр, создание и решение задачи, механизмы менторинга).

Для создания клиент-серверного приложения Learn_JS использовалась кроссплатформенная среда разработки WebStorm. Для хранения информации о тестах применялась документоориентированная система управления базами данных MongoDB, считающаяся одним из классических примеров NoSQL-систем и использующая JSON-подобные документы. Клиентская часть приложения разработана с использованием библиотеки создания пользовательских интерфейсов React, а серверная часть – с применением фреймворка NestJS.

Были разработаны две основные модели, в декларационном стиле описывающие структуру сохраняемого документа, реализованные в базе данных MongoDB: задача и тест. Модель задачи представлена на рисунке 1.

```
const taskSchema: Schema = new Schema( definition: {
    name: {
        type: String,
        required: true,
        unique: true
    },
    description: {
        type: String,
        required: true
    },
    createdBy: {
        type: Types.ObjectId,
        required: true,
        ref: 'Users'
    },
    createdDate: {
        type: Date,
        default: Date.now()
    },
    averageRating: {
        type: Number,
        default: 0
    },
    numberOfRatingReviews: {
        type: Number,
        default: 0
    },
    averageDifficult: {
       type: Number,
       default: 0
    },
    numberOfDifficultReviews: {
        type: Number,
        default: 0
    },
});
```

Рисунок 1 – Модель задачи

Задача имеет следующие поля: name – обязательное и уникальное текстовое поле с названием задачи; description – обязательное текстовое поле с описанием задачи (постановка задачи); createdBy – обязательное поле с типом ObjectId, ссылающееся на документ пользователя-создателя задачи во внешней таблице с пользователями;

createdDate – поле с типом дата, содержит значение по умолчанию – дату создания задачи; averageRating – числовое поле, содержащее текущий рейтинг задачи, значение по умолчанию – 0; numberOfRatingReviews – числовое поле, содержащее количество оценок рейтинга, значение по умолчанию – 0; averageDifficult – числовое поле, содержащее среднее значение сложности задачи, значение по умолчанию – 0, используется для пересчета рейтинга задачи; numberOfDifficultReviews – числовое поле, содержащее количество оценок сложности задачи, значение по умолчанию – 0, используется для пересчета сложности задачи.

Модель теста представлена на рисунке 2.

```
const testSchema: Schema = new Schema( definition: {
    taskId: {
        type: Types.ObjectId,
        ref: 'Tasks',
        required: true
    },
    input: {
        type: {},
        required: true
    },
    output: {
        type: {},
        required: true
    },
    isExample: {
        type: Boolean,
        default: true
    }
});
```

Рисунок 2 – Модель теста

Тест имеет следующие поля: taskId – обязательное поле с типом ObjectId, ссылающееся на документ задачи во внешней таблице с задачами; input – обязательное поле с типом Mixed, способное содержать данные любого формата, содержит входные данные для задачи; output – обязательное поле с типом Mixed, способное содержать данные любого формата, содержит выходные данные для задачи; isExample – булево поле, определяющее, является ли тест презентационным (возможно ли его показать пользователю) со значением по умолчанию – true. Отношение коллекции «Задачи» к коллекции «Тесты» можно интерпретировать как один ко многим, т. е. одна задача может иметь множество тестов.

Разработанное приложение Learn_JS позволяет автоматически проверять правильность выполнения задания обучающимся. Пошаговая реализация процесса проверки решения задачи приложением Learn_JS, после того как пользователь ее решит и отправит свое решение в виде функции, которое поступает на обработку в соответствующий контроллер на сервере приложения, приведена ниже.

1) Будут найдены все тесты со значением поля tasked, таким же как id задачи, поступившей на решение, в соответствующей таблице.

2) На основе пришедшего на тестирование текстового представления будет создана новая функция путем транслирования на язык программирования JavaScript.

3) Для каждого теста функция получит исходные параметры теста в качестве входных параметров функции, далее сравнит полученный результат выполнения с ожидаемым результатом, который взят из БД тестов.

4) Если задача прошла тест, количество пройденных тестов будет увеличено на 1.

5) Если задача не прошла тест, количество непройдённых тестов будет увеличено на 1, и если тест помечен как доступный для демонстрационного примера, его входные и выходные значения будут добавлены в ответ сервера, который вернется на сторону клиента и будет показан пользователю.

6) После того, как решение будет протестировано всеми тестами, вернется ответ, содержащий флаг, является ли решение правильным, информацию о количестве пройденных тестов, количестве непройденных тестов, а также входных и выходных данных непройденных тестов, которые доступны для показа пользователю.

Так как в модели документа теста задачи входные и выходные данные имеют тип Mixed, то имеется возможность создания большого числа задач и тестов для них из-за отсутствия регламента на данные тестов. В качестве данных могут использоваться числа, массивы, объекты и т. д.

В перспективе развития приложения Learn_JS возможны добавление теоретической части, добавление соответствующих теоретических тем к задаче; усовершенствование системы оценивания сложности задачи; фильтрация доступных задач, основываясь на необходимых теоретических темах, сложности и рейтинге задачи; добавление механизмов совместной разработки в режиме реального времени для парного программирования студента и его ментора.

Обучающее клиент-серверное приложение Learn_JS по изучению языка программирования JavaScript прошло проверку первыми обучающимися, которые попользовались приложением и отметили следующие достоинства и недостатки. Приложение доступно для апробации по адресу https://javascript-learn.herokuapp.com/.

Достоинства: 1) возможность самостоятельного обучения; 2) постоянное пополнение базы задач; 3) возможность взаимодействия с ментором (назначает задачи, чтобы восполнить пробелы и проработать проблемные темы, можно обратиться с просьбой о помощи); 4) удобный онлайн-редактор кода с цветовым выделением ключевых слов.

Недостатки: 1) отсутствие теоретического раздела в приложении; 2) отсутствие возможности фильтровать задачи по темам, рейтингу и сложности; 3) отсутствие возможности парного программирования (для совместной работы с ментором).

Исходный код приложения можно посмотреть по ссылке https://github.com/Zaytsev-Alex/learn-js/tree/master.

Литература

1 Зайцев, А.С. Разработка обучающего клиент-серверного приложения по изучению языка программирования Javascript / А. С. Зайцев // Творчество молодых’ 2021 : сборник научных работ студентов, магистрантов и аспирантов : в 3 ч. / ГГУ им. Ф. Скорины; Р. Б. Бородич (гл. ред.) [и др.]. – Гомель, 2021. – Ч. 1. – С. 198–201.

2 Браун, Э. Изучаем JavaScript: руководство по созданию современных веб-сайтов / Э. Браун. – Изд. 3-е. – Санкт-Петербург : Диалектика, 2017. – 368 с.

УДК 004.91

Д. А. Иокша

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ СБОРА ПЕРИОДИЧЕСКОЙ И РАЗОВОЙ ОТЧЕТНОСТИ С РЕГИОНАЛЬНЫХ ПРЕДСТАВИТЕЛЕЙ ОРГАНИЗАЦИИ

Статья посвящена автоматизации сбора периодической и разовой отчетности с региональных представителей организации. Рассмотрены способы организации и архитектуры программного обеспечения, навигации между страницами и компонентами приложения. В статье изложены требования к функционалу приложения со стороны