DOI 10.37882/2223-2966.2024.8.25

РАЗРАБОТКА WEB-ПРИЛОЖЕНИЯ ДЛЯ ВИЗУАЛЬНОГО ПРОЕКТИРОВАНИЯ СХЕМЫ ДАННЫХ С ВОЗМОЖНОСТЬЮ ВЫГРУЗКИ СКРИПТА ДЛЯ СОЗДАНИЯ ТАБЛИЦ

DEVELOPMENT OF A WEB APPLICATION FOR VISUAL DESIGN OF A DATA SCHEME WITH THE POSSIBILITY OF UPLOADING A SCRIPT FOR CREATION OF TABLES

E. Prokhorov
A. Dorofeev

Summary. The article is devoted to the issue of developing a web application for designing a data schema with the ability to generate and download an SQL script to create data tables of the selected DBMS and SQL language dialect. The architecture of the web application, the technology stack for its implementation is considered, and the general interface of the system is presented. The planned inclusion of a subsystem for setting the correspondence of table creation operators and setting attribute characteristics using the theory of translators and setting grammar and rules for a specific SQL dialect will expand the capabilities of the created application.

Keywords: data schema, relational database, SQL, script generation, web application, PosgreSQL, Node.js, React.js.

Прохоров Евгений Викторович
Иркутский национальный исследовательский технический университет
Cherem2002@gmail.com
Дорофеев Андрей Сергеевич
Кандидат технических наук, доцент,
Иркутский национальный исследовательский технический университет
dorbaik@ex.istu.edu

Аннотация. Статья посвящена вопросу разработки web-приложения для проектирования схемы данных с возможностью генерации и выгрузки SQL-скрипта для создания таблиц данных выбранных СУБД и диалекта языка SQL. Рассматривается архитектура web-приложения, стек технологий для его реализации, представлен общий интерфейс системы. Планируемое включение подсистемы настройки соответствия операторов создания таблиц и задания характеристик атрибутов с использованием теории трансляторов и задания грамматики и правил для конкретного диалекта SQL расширит возможности созданного приложения.

Ключевые слова: схема данных, реляционная база данных, SQL, генерация скрипта, web-приложение, PosgreSQL, Node.js, React.js.

Введение

В современном мире разработка приложений для работы с базами данных играет ключевую роль в обеспечении эффективного управления информацией. Язык SQL (Structured Query Language) является одним из основных инструментов для работы с реляционными базами данных, которые до сих пор являются самыми распространенными, позволяя разработчикам создавать, изменять, извлекать и управлять данными [1]. Сегодня SQL — самый популярный и широко используемый язык запросов в мире, применяемый в различных областях, включая веб-разработку, аналитику данных, бизнес-анализ, финансы, медицину и многое другое. В современном мире SQL также стал интегрироваться с новыми технологиями, такими как машинное обучение и искусственный интеллект, что позволяет использовать его для анализа и обработки больших объемов данных, выявления паттернов и предсказания будущих событий. Таким образом, SQL продолжает эволюционировать, адаптируясь к изменяющимся потребностям и технологическим трендам, и остается одним из ключевых инструментов в мире информационных технологий.

Создание скриптов для создания баз данных упрощает процесс развертывания проекта на новых серверах или его миграции на другие базы данных. Путем выполнения скриптов можно быстро восстановить структуру базы данных в новом окружении. Скрипты для создания баз данных также могут служить документацией к проекту, описывая его структуру и отношения между данными. Это упрощает понимание проекта другими разработчиками и членами команды.

1. Постановка задачи

Разработка SQL-скриптов может быть достаточно нетривиальной и трудоемкой задачей, особенно при работе с большими и сложными схемами баз данных. На помощь разработчикам приходят различные CASE-средства и приложения, применяемые для проектирования баз данных. Здесь следует отметить DBDesigner [2], MySQL Workbench [3], Valentina Studio [4] и многие другие. У всех них есть, естественно, свои плюсы и минусы. Однако, на наш взгляд, недостатком является явная ориентация на определенные СУБД, их диалекты, генерация скрипта только под выбранную (-ые) СУБД.

В этой связи становится актуальной задача разработки и дальнейшего развития инструмента, позволяющего визуально построить схему данных, а затем получать скрипт по настроенным в этой же системе правилам.

Основными преимуществами разрабатываемого приложения должны стать: использование свободно-распространяемого программного обеспечения при проектировании и разработке, открытый исходный код, возможность генерировать SQL-код, поддержка различных диалектов SQL, интуитивно понятный интерфейс.

2. Применяемые средства разработки

Backend, или серверная часть, представляет собой основную часть веб-приложения, ответственную за обработку запросов от клиентской части (frontend) и взаимодействие с базами данных, внешними сервисами и другими компонентами системы [5]. Он обеспечивает логику приложения, обработку данных, аутентификацию и авторизацию пользователей, а также бизнес-логику, необходимую для функционирования приложения. Выбор технологий для backend-разработки играет ключевую роль в определении производительности, масштабируемости, безопасности и функциональности веб-приложения.

Frontend, или клиентская часть, представляет собой интерфейс веб-приложения, с которым взаимодействуют пользователи. Он отвечает за отображение данных, интерактивность и визуальное взаимодействие пользователя с приложением. Выбор технологий для frontend-разработки имеет решающее значение для создания удобного, функционального и привлекательного пользовательского интерфейса.

Ключевые факторы, которые следует учитывать при выборе технологий для frontend-разработки, включают в себя производительность, масштабируемость, поддержку современных стандартов веб-разработки, а также удобство использования и обучения для разработчиков.

Выбор авторов пал на среду выполнения JavaScript-программ Node.js [6] для backend, который отлично подходит для создания быстрых и масштабируемых веб-приложений с высокой производительностью и удобством разработки, и React.js [7], представляющий собой мощный инструмент для создания современных веб-приложений, обладающих высокой производительностью, гибкостью и широкой поддержкой сообщества. Для хранения данных разрабатываемого приложения используется СУБД PosgreSQL, являющаяся наиболее развивающейся в настоящее время, которая подходит для крупных и сложных проектов, требующих богатый набор функциональных возможностей и высокую надежность [8]. Инструментами разработки стали pgAdmin, Visual Studio Code (VS Code), Git — распределенная система управления версиями.

3. Проектирование базы данных и общая архитектура приложения

Модель данных — это совокупность правил порождения структур данных в базе данных, операций над ними, а также ограничений целостности, определяющих допустимые связи и значения данных, последовательность их изменения [9].

На рис. 1 представлена обобщенная физическая модель данных реализуемого приложения для СУБД PostgreSQL.

Разрабатываемое программное средство представляет собой современное веб-приложение с использованием технологий Node.js, Express.js, React.js и PostgreSQL. Оно разделено на серверную и клиентскую части, каждая из которых имеет свою структуру и зону ответственности (рис. 2).

Основными подсистемами приложения являются следующие:

— подсистема управления данными, включающая в себя все функции, связанные с получением, обработкой, изменением и хранением данных, таких как таблицы, записи и запросы к базе данных;

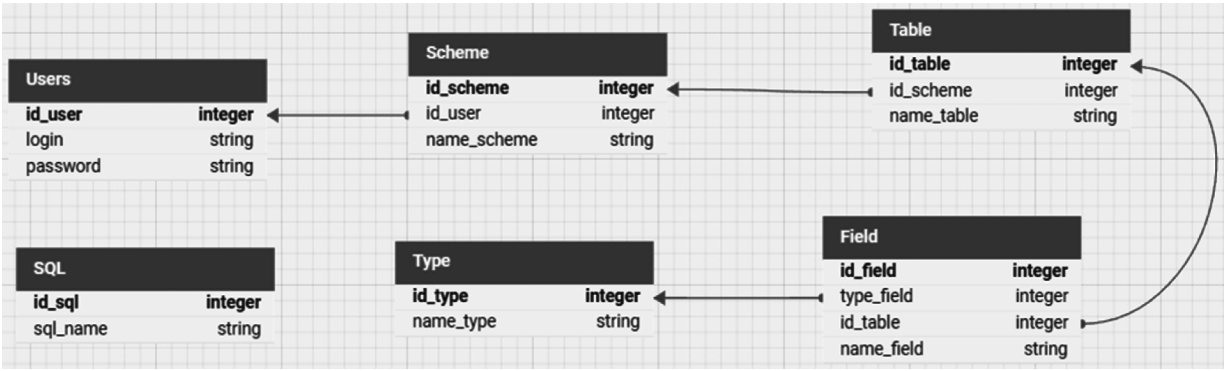

Рис. 1. Обобщенная физическая модель данных приложения

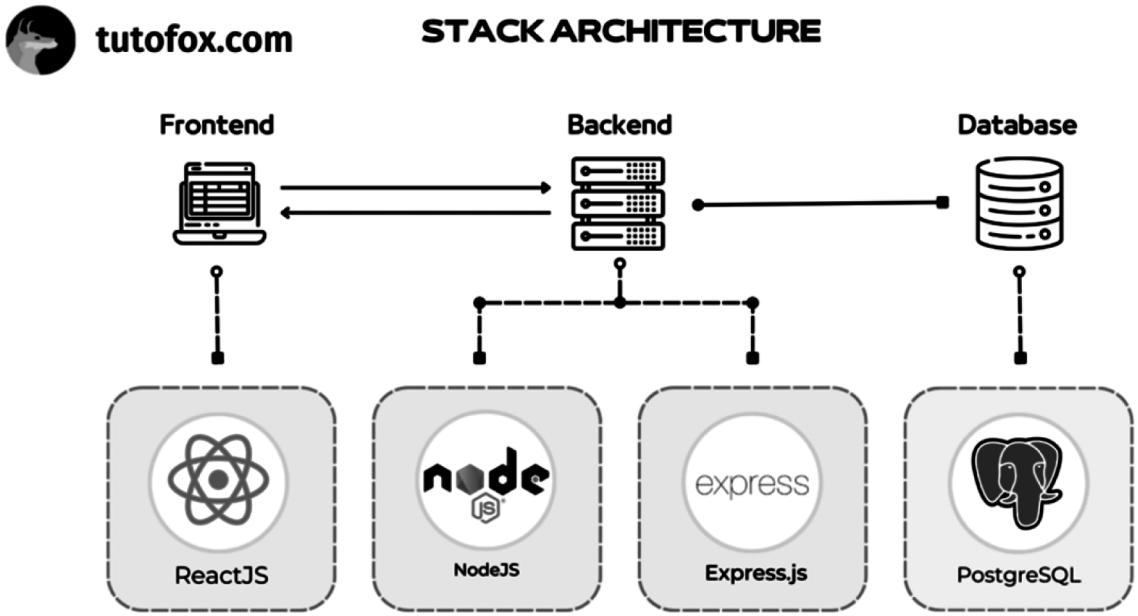

Рис. 2. Архитектура системы

— подсистема пользовательского интерфейса (UI), отвечающая за отображение данных и взаимодействие с пользователем через интерфейс пользователя;
— подсистема аутентификации и авторизации;
— подсистема обработки данных, включающая в себя все функции и алгоритмы обработки данных, такие как фильтрация, сортировка, преобразование и т. д.;
— подсистема экспорта и импорта данных, отвечающая за возможность экспорта данных из приложения во внешние форматы и импорта данных из внешних источников.
— подсистема взаимодействия с внешними сервисами и API, ответственная за взаимодействие с внешними сервисами, API и другими компонентами, которые могут предоставлять дополнительную функциональность или данные для приложения.

Краткое описание модулей приложения и их иерархия представлены на рис. 3 и в таблице 1.

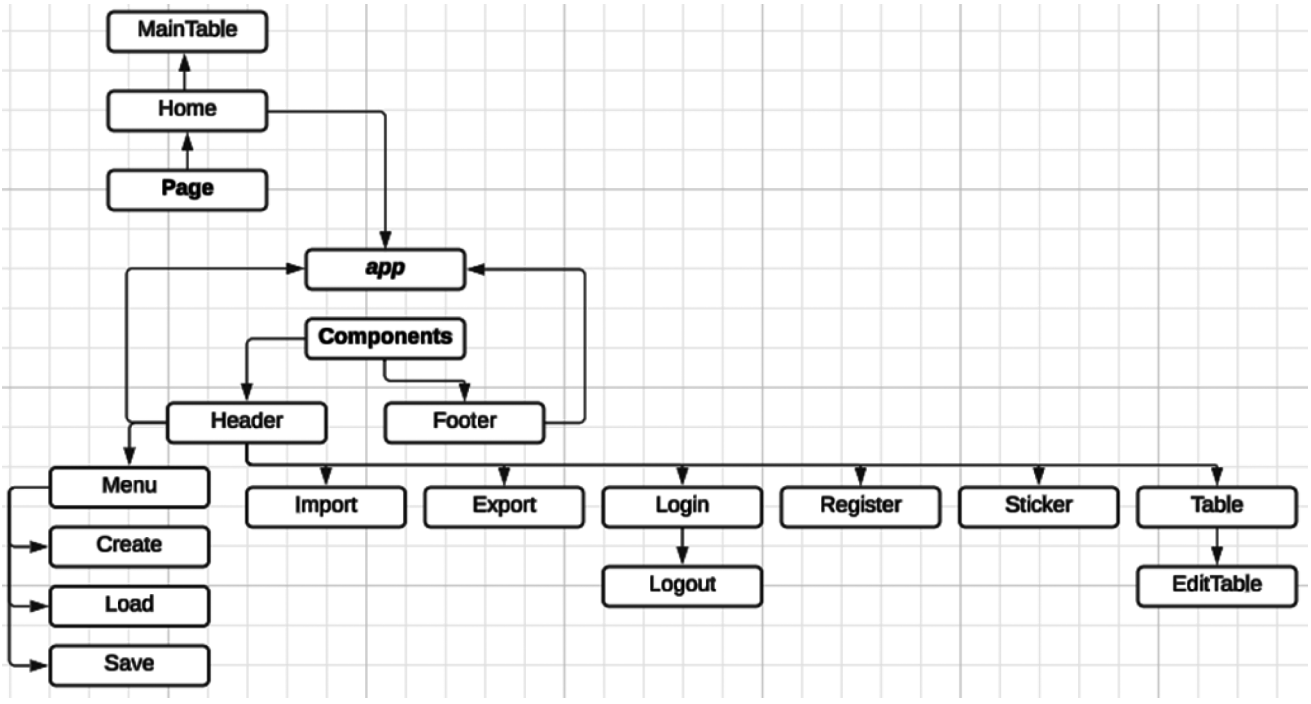

Рис. 3. Связь модулей приложения

Таблица 1.

Функции модулей

Название	Назначение	Тип
Header	Компонент предназначен для отображения верхней части приложения. Он содержит такие элементы, как название, навигационное меню, ссылки на основные разделы приложения	Component
Menu	Навигационное меню — это часть веб-интерфейса, предназначенная для навигации пользователя по различным разделам приложения.	
Create	Компонент, где мы можем ввести название схемы.	
Load	Компонент для загрузки одной из схем (в случае если пользователь авторизован)	
Save	Компонент для сохранения изменений в схеме	
Import	Компонент для импорта схемы	
Export	Компонент для экспорта схемы	
Login	Компонент для входа	
Logout	Компонент для выхода	
Register	Компонент для регистрации	
Sticker	Компонент для размещения заметок	
Table	Компонент для создания таблицы	
EditTable	Компонент для редактирования таблицы	
MainTable	Компонент для отображения таблицы на рабочем пространстве	
Home	Главная страница	Page

В React.js page (страница) обычно представляет собой компонент, который отображает содержимое конкретной страницы веб-приложения. Каждая страница может содержать уникальный URL, различное содержимое и логику, и может быть отображена в браузере как отдельная веб-страница. Component (компонент) представляет собой многократно используемый «строительный блок» интерфейса, который объединяет в себе HTML, CSS и JavaScript для создания отдельных частей пользовательского интерфейса.

4. Интерфейс web-приложения

Рассмотрим интерфейс приложения. После успешной авторизации пользователя ему становится доступно основное окно проектирования схемы данных. На рис. 4 представлен интерфейс главной страницы приложения с построенной схемой данных, состоящей из нескольких взаимосвязанных таблиц.

Пример окна создания таблицы, ее полей, первичного и внешних ключей, задания автоинкрементного поля, уникальных атрибутов представлен на рис. 5.

На рис. 6–7 изображено содержимое файлов с генерированным скриптом для двух выбранных для примера СУБД.

Результаты тестирования позволяют сделать вывод о корректности работы разработанного web-приложения.

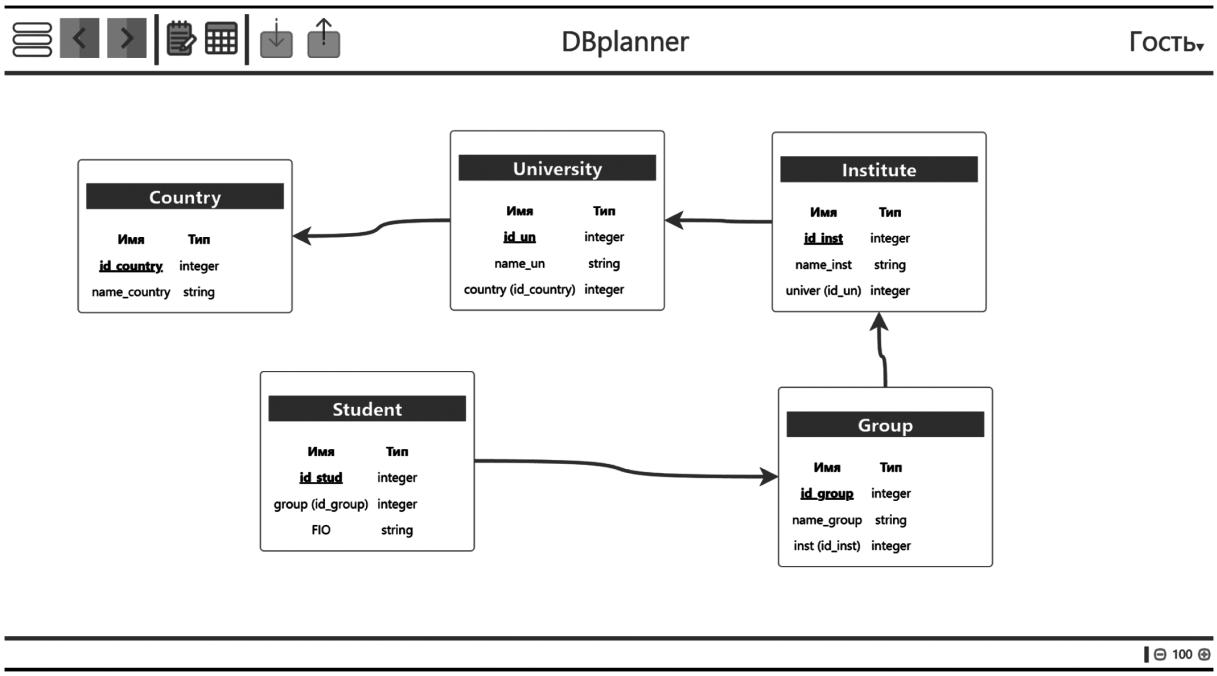

Рис. 4. Общий вид приложения

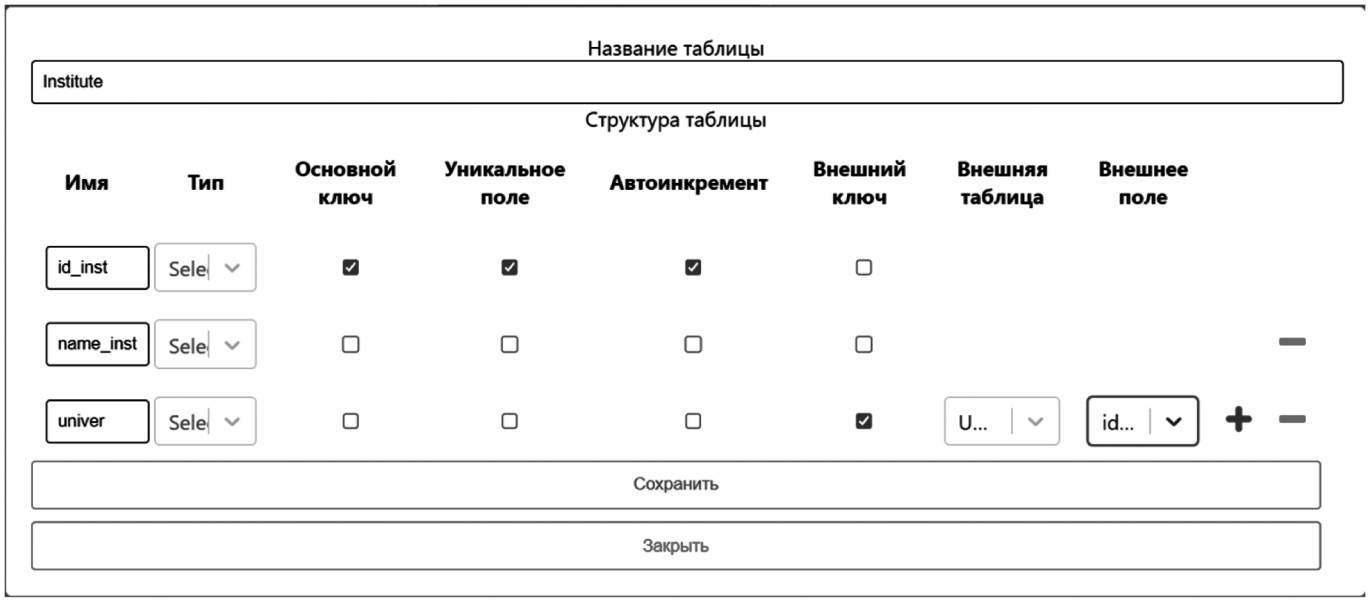

Рис. 5. Атрибуты таблицы

```sql
CREATE TABLE Country (
    id_country integer PRIMARY KEY UNIQUE IDENTITY(1,1),
    name_country string
);

CREATE TABLE University (
    id_un integer PRIMARY KEY UNIQUE IDENTITY(1,1),
    name_un string,
    country integer, FOREIGN KEY (country) REFERENCES Country(id_country)
);

CREATE TABLE Institute (
    id_inst integer PRIMARY KEY UNIQUE IDENTITY(1,1),
    name_inst string,
    univer integer, FOREIGN KEY (univer) REFERENCES University(id_un)
);

CREATE TABLE Group (
    id_group integer PRIMARY KEY UNIQUE IDENTITY(1,1),
    name_group string,
    inst integer, FOREIGN KEY (inst) REFERENCES Institute(id_inst)
);

CREATE TABLE Student (
    id_stud integer PRIMARY KEY UNIQUE IDENTITY(1,1),
    group integer, FOREIGN KEY (group) REFERENCES Group(id_group),
    FIO string
);
```

Рис. 6. Скрипт для MySQL

Заключение

Представленная система является первым вариантом средства для построения модели данных с возможностью получения сгенерированного скрипта для создания таблиц базы данных для диалектов различных СУБД, которые в настоящее время ограничены Oracle, MySQL, и PostgreSQL и MS SQL. Планируется реинжиниринг схемы данных по загруженному скрипту, расширение предоставляемых систем управления базами данных, а также разработка подсистемы настройки соответствия операторов создания таблиц и задания характеристик атрибутов с использованием теории трансляторов и задания грамматики и правил для конкретного диалекта SQL.

```
CREATE TABLE Country (
    id_country integer PRIMARY KEY UNIQUE SERIAL,
    name_country string
);

CREATE TABLE University (
    id_un integer PRIMARY KEY UNIQUE SERIAL,
    name_un string,
    country integer, FOREIGN KEY (country) REFERENCES Country(id_country)
);

CREATE TABLE Institute (
    id_inst integer PRIMARY KEY UNIQUE SERIAL,
    name_inst string,
    univer integer, FOREIGN KEY (univer) REFERENCES University(id_un)
);

CREATE TABLE Group (
    id_group integer PRIMARY KEY UNIQUE SERIAL,
    name_group string,
    inst integer, FOREIGN KEY (inst) REFERENCES Institute(id_inst)
);

CREATE TABLE Student (
    id_stud integer PRIMARY KEY UNIQUE SERIAL,
    group integer, FOREIGN KEY (group) REFERENCES Group(id_group),
    FIO string
);
```

Рис. 7. Скрипт для PostgreSQL

ЛИТЕРАТУРА

1. Грофф, Джеймс Р., Вайнберг, Пол Н., Оппелъ, Эндрю Дж. SQL: полное руководство: Пер. с англ. — М.: ООО «И.Д. Вильяме», 2015. — 960 с.
2. DBDesigner [Электронный ресурс] URL: https://www.dbdesigner.net (дата обращения: 15.05.2024).
3. MySQL Workbench [Электронный ресурс] URL: https://www.mysql.com/products/workbench/. (дата обращения: 20.05.2024).
4. Valentina Studio [Электронный ресурс] URL: https://www.valentina-db.com/ (дата обращения: 20.05.2024).
5. Testengineer.ru [Электронный ресурс] URL: https://testengineer.ru// (дата обращения: 25.05.2024).
6. Попова, Ю.Ю. Node.js: разработка приложений в микросервисной архитектуре с нуля / Ю.Ю. Попова. — Санкт-Петербург: БХВ-Петербург, 2024. — 256 с.
7. Бэнкс Алекс, Порселло Ева. React: современные шаблоны для разработки приложений / А. Бэнкс, Е. Порселло. — Санкт-Петербург: Питер, 2022. — 320 с.
8. PostgreSQL Documentation. [Электронный ресурс]. URL: https://www.postgresql.org/docs/ (дата обращения 17.05.2024).
9. Даталогические модели данных [Электронный ресурс] URL: https://spravochnaya.com/7514_datalogicheskie-modeli-dannyh.html (дата обращения: 10.06.2024).

Журнал «Современная наука: актуальные проблемы теории и практики»