Ю. В. Соловей, С. В. Столяров, М. С. Воробьева, М. В. Аврискин
Тюменский государственный университет, г. Тюмень
УДК 004.2

РАЗРАБОТКА АРХИТЕКТУРЫ WEB-СЕРВИСА ДЛЯ УЧЕТА СТУДЕНЧЕСКИХ ИТ-ПРОЕКТОВ

Аннотация. Для студентов затруднителен доступ к ИТ-проектам, реализованным во время учебных дисциплин. В статье описывается разработка архитектуры Web-сервиса для учета учебных проектов, предоставляющего доступ к ним и инструменты поиска. Сервис включает Django-сервер для работы с проектами, Git-север Gitea, сервер аутентификации Keycloak. Рассматриваются преимущества и недостатки хранения файлов проектов в базе данных.
Ключевые слова: ИТ-проект, хранение проектов, Web-сервис, хранение файлов, Gitea, Django, OIDC.

Введение. Во время обучения студент выполняет проекты, в качестве которых может выступать типовая задача или задача, поставленная на решение актуальной проблемы. Темы в проектах могут повторяться со временем, что для типовых задач нормально, так как студенческие проекты направлены на изучение и закрепление материала, а для задач, поставленных на решение актуальной проблемы, все не так однозначно — работа, не вносящая новых решений в проблему, теряет практическую ценность. Тем не менее, если студент учитывает результаты других проектов, то это может приблизить задачу к своему решению, а также послужить для него хорошим опытом.

На данный момент разработанные студенческие проекты хранятся у преподавателей на личных хранителях, а также в виде физических копий. Для студента нет возможности посмотреть список проектов, затруднительно найти интересующий проект, так как информацию о наличии или о проектах с похожей тематикой могут знать не все преподаватели. Для упрощения процесса получения и поиска проектов, а также для формирования цифрового следа студентов решено организовать общий доступ к ним и предоставить набор инструментов для поиска [1]. При этом формирование цифровой репутации является актуальной задачей, в которой заинтересован как студент, так и преподаватель [2].

Целью данной работы является проектирование и разработка архитектуры Web-сервиса, который позволит пользователям получить доступ к студенческим проектам как для ознакомления с результатами решения, так и для возможного продолжения проекта.

Построение системы включает в себя использование шаблона проектирования MVC [3] для разработки Web-сервера с использованием фреймворка Django [4] и стандарта OpenID Connect [5] при формировании микросервисной архитектуры.

Методы и технологии. В процессе образовательной деятельности студенты выполняют проекты, в рамках которых формируется его цифровой след. Список документов и данных, которые формирует студент в рамках проектной деятельности, зависит как от дисциплины, в рамках которой проводится проект, так и от специфики проекта. Для ИТ-проектов в качестве результата характерно наличие программного кода, используемых ресурсов и сопроводительных документов (инструкция пользователя, техническое задание, инструкция по установке и т. д.).

Поэтому для формирования информационного хранилища была определена структура данных, которая включает обязательные, персонализированные и опциональные характеристики проекта. К обязательным относятся: краткое и полное название, аннотация, краткое описание актуальности и решенных задачах, итоговый отчет, презентация, дата публикации. К персонализированным параметрам относится: состав команды студентов, выполнивших проект, курс и направление, дисциплина и преподаватель. Опциональные характеристики — это программный код, датасет и технические документы.

Для каждой группы параметров определен разный способ хранения входных данных разных форматов: структурированные и неструктурированные данные, метаданные, файловые данные. Структурированные и метаданные хранятся в реляционной базе данных, документы — в файловом хранилище, программный код — в системе управления версий.

Для разработки главного Web-сервера использовался фреймворк Django языка Python. Верстка Web-интерфейса страниц Django сервера произведена с использованием приложения Bootstrap Studio в

связке с проектом botstrap-studio-to-django-template, предназначенным для преобразования сгенерированных страниц в файлы шаблонов Django. Для управления ролями и идентификацией произведена настройка сервера Keycloak, который позволяет использовать технологию единого входа SSO, для авторизации пользователей используется протокол OpenID Connect.

В качестве сервиса для хранения программного кода студенческих проектов был выбран open source git-сервис Gitea, взаимодействие с Api-интерфейсом сервисов Gitea и Keycloak выполнялось с помощью библиотек Requests и mozilla-django-oidc.

Для индексации текстовых данных и поиска проектов использовалась библиотека Whoosh. Индексирование помогает выразить содержание текста документа в терминах языка информационно-поисковой системы. Также с помощью инвертированного индекса затрачиваемое время на обновление индекса меньше, чем при повторном анализе текста.

Для хранения структурированных данных используется реляционная база данных PostgreSQL. Обратный прокси-сервер Nginx используется для перенаправления http-вызовов пользователей к различным сервисным компонентам. Для автоматизации развертывания проекта используется технология контейнеризации Docker.

При разработке использовалась архитектура MVC (Model View Controller), которая эффективно отражает рабочие процессы, бизнес-логику и взаимодействие с пользователем и его ожидаемые результаты, совмещает простоту разработки системы и удобство использования [6].

В рамках исследования возникла задача выбрать способ хранения отчетных документов. Рассматривались варианты хранения в реляционной базе данных в виде BLOB поля или в файловом хранилище. При этом необходимые преимущества, которые дает хранение в базе данных, а именно атомарность операций, возможность скрыть название при хранении, разграничение доступа к ресурсу и совместное хранение с метаданными могут быть получены при хранении информации о местонахождении файла в базе данных. Из-за более высокой скорости чтения и записи для хранения файлов больше

256 кб выгоднее использовать файловую систему для хранения файлов. Особенно в долгосрочной перспективе из-за фрагментации [7].

Для оценки времени работы на реальных данных был произведен замер скорости загрузки файлов выпускных квалификационных работ выпускников направления МОиАИС 2022 года. Результат был усреднен за счет большого количества повторений эксперимента. В общей сложности каждый из 31 собранного файла был 63 раза загружен в каждое из хранилищ, была произведена сравнительная оценка времени работы. Запись файлов в файловую систему потребовало в 2.5 раз меньше времени. Оценка происходила на компьютере с использованием процессора AMD 5500U и 16 GB оперативной памяти.

Чтобы совершать поиск по загруженным проектам, для анализа были взяты обязательные данные. Пусть есть множество проектов P = $\{P_1, P_2, P_3, .. P_n\}$, где каждый документ p ∈ P имеет следующие атрибуты: название (текстовое значение до 100 символов), краткое описание (текстовое значение до 1000 символов), итоговый отчет (документ в формате *.pdf или *.doc). Пользовательский запрос q состоит из ключевых слов и фраз, а также есть набор фильтров F, где каждый фильтр f ∈ F представляет собой условие, которому должны удовлетворять документы.

Задача поиска состоит в нахождении подмножества документов R ⊆ P, которое максимизирует функцию релевантности Score(p, q) для каждого документа p ∈ R при условии удовлетворения набора фильтров F.

$$R = argmax(Score(p, q)), p \in P$$

$$subject\ to: Filter(p, F) = true, p \in R,$$

где Score(p, q) — функция релевантности, которая оценивает степень соответствия документа p запросу q;

Filter(p, F) — функция фильтрации, которая проверяет, удовлетворяет ли документ p набору фильтров F, это может быть логическое выражение, которое оценивается как true, если документ p удовлетворяет всем фильтрам из F, и false в противном случае.

Описание архитектуры. Архитектура Web-сервиса представляет собой набор компонентов, которые отвечают за работу с определенными данными и запущены в контейнере Docker. Внутреннее взаимодействие с другими компонентами происходит только посредством http-запросов внутри локальной сети Docker Compose. Главным компонентом является Django-сервер, отвечающий за загрузку данных проекта на сервер, логику создания, проведения проектных марафонов, страницы для поиска и просмотра опубликованных проектов. Между Git-сервером и Django-сервером происходит взаимодействие через Api-запросы для получения и редактирования информации о репозиториях пользователей. Для идентификации пользователя через внешний сервис аутентификации компоненты Python Server и Gitea используют поток Authorization Code Flow, так как клиентское приложение находится на удаленном сервере и имеет возможность держать в сохранности секретный ключ.

Схема заполнения хранилища данных включает несколько уровней (рис. 1). Загружаемые пользователем документы архивируются для сокращения занимаемого места на диске, при этом итоговый отчет по проекту перед архивацией индексируется, чтобы обеспечить поиск по содержимому.

За формирование и обновление состояния индекса в файловом хранилище отвечает поисковая система, использующаяся на сервере. Программный код проекта публикуется на Git-сервере, в роли которого выступает Gitea. Помимо структурированных данных и метаданных, введенных пользователем, в базу данных поступает информация о состоянии репозиториев из компонента Gitea.

При работе с системой пользователь взаимодействует с несколькими функциональными компонентами (рис. 2). Использование Python сервера происходит для выполнения различных функций таких как поиск, загрузка, скачивание проектов, управление дисциплинами и проектными марафонами. Компонент Gitea предоставляет возможность работы с репозиториями кода, созданием, редактированием. Компонент Keycloak позволяет получить доступ к другим компонентам системы обеспечивает получение токена аутентификации.

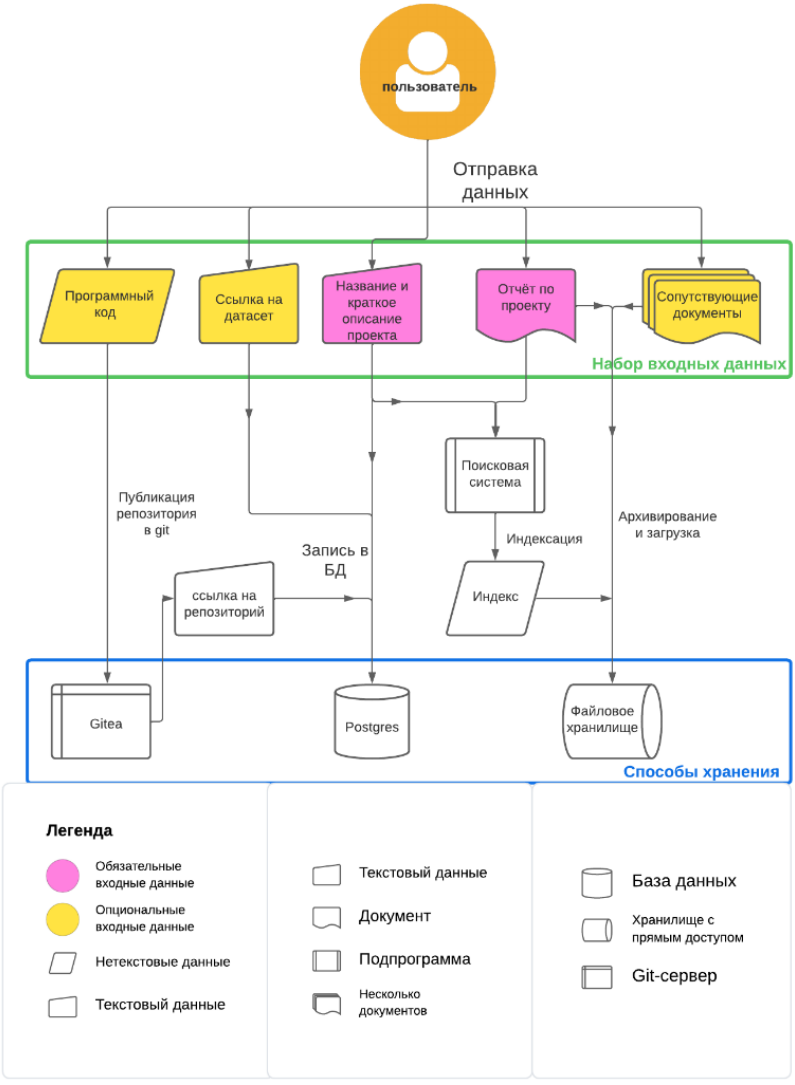

Рис. 1. Схема обработки и хранения данных

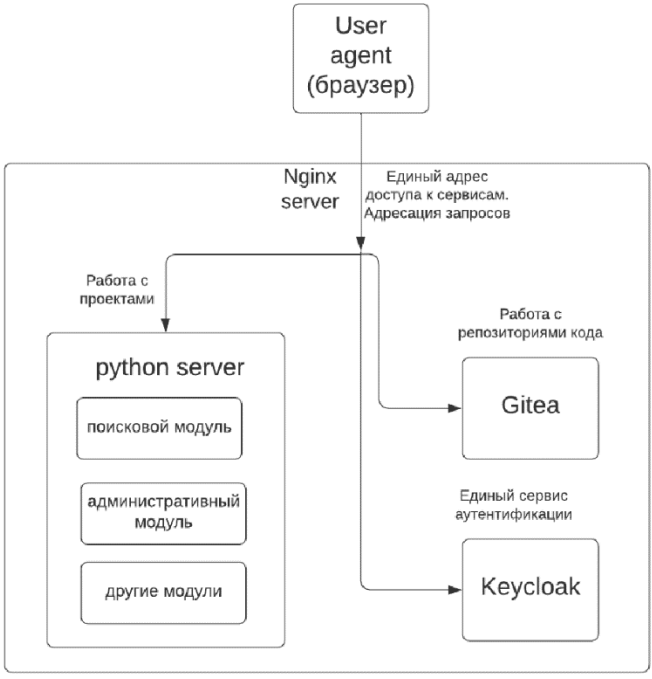

Рис. 2. Функциональное взаимодействие пользователя с компонентами сервиса

Python сервер включает в себя несколько модулей, которые обеспечивают взаимодействие с другими компонентами (рис. 3). Один из таких модулей предназначен для работы с Api-интерфейсом Gitea и реализует функционал, который позволяет получать списки репозиториев, определять используемые в них языки программирования и изменять характеристики репозитория.

Другой модуль отвечает за обработку Webhook уведомлений, которые позволяют отслеживать события, связанные с созданием, удалением или переименованием репозиториев на сервисе Gitea.

Модуль идентификации отвечает за получение пользовательских identity и токена аутентификации через компонент Keycloak.

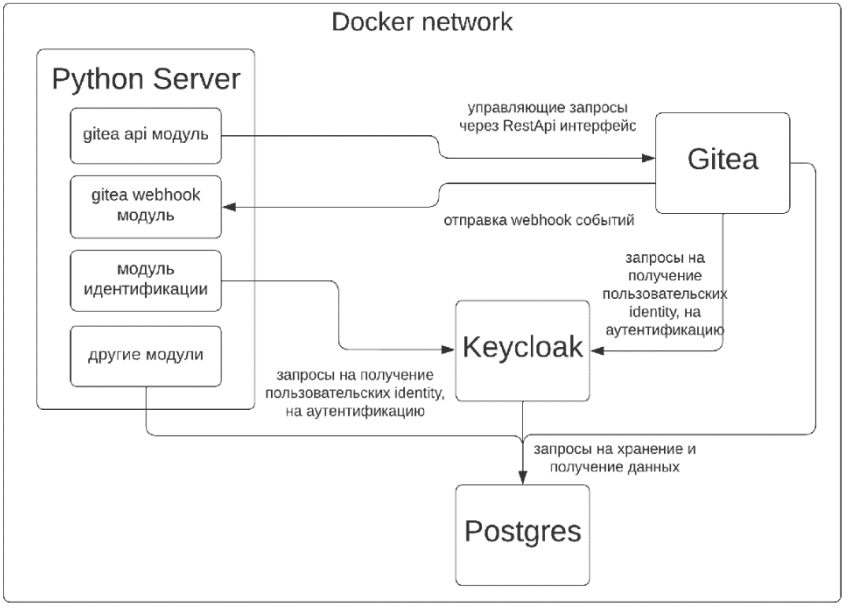

Рис. 3. Диаграмма внутреннего взаимодействия компонентов сервиса

Результаты. С учетом построенной архитектуры был разработан и запущен Web-сервис для хранения студенческих ИТ-проектов ИМиКН ТюмГУ. Полная схема публикации проекта в системе — это процесс создания публичного проекта, описанного от лица нескольких пользователей: руководителя проектной дисциплины, создателя проекта и научного руководителя (рис.4).

Преподаватель, создающий дисциплину, становится руководителем, после добавления участников курса, студенты получают возможность загружать проекты. В процессе проектной деятельности студенты имеют возможность пригласить соруководителей и дополнительных участников проекта.

По окончанию загрузки проектов студенты запрашивают доступ на публикацию у научного руководителя. После подтверждения проект получает статус “опубликованный” и доступен всем участникам Web- сервиса.

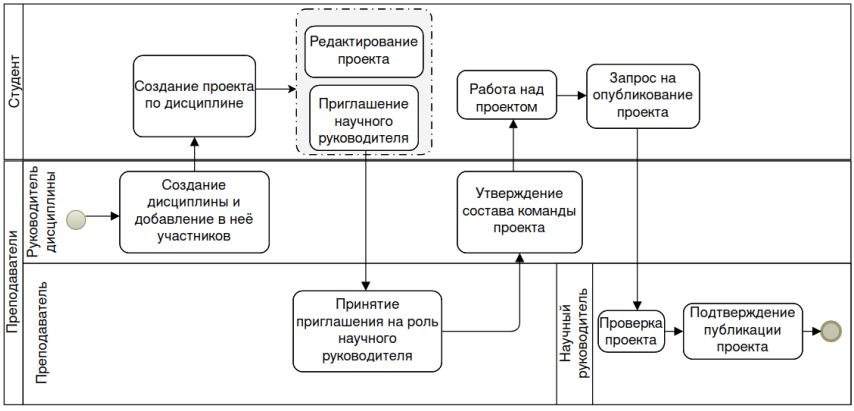

Рис. 4. Схема процесса создания и публикации проекта в системе

Система была развернута на виртуальном сервере ТюмГУ с 10 виртуальными ядрами Intel X5670, 20GB оперативной памяти и диском объемом 500 + 50 GB SSD.

Сервис был протестирован, загружены 53 проектов студентов 3 и 4 курса направления МОиАИС по нескольким дисциплинам («Методы и технологии машинного обучения», «Разработка систем обработки данных в предметных областях», «Методы и средства проектирования пользовательского интерфейса», «Технологическая практика», «Интеллектуальные системы поддержки принятия решений», «Структуры и алгоритмы компьютерной обработки данных»).

Проведенный сбор отзывов пользователей позволил выявить неточности и улучшить логику системы до его запуска в постоянную эксплуатацию. Такая практика является эффективным инструментом для повышения качества ИТ-проектов.

Заключение. Разработан Web-сервер, позволяющий хранить данные студенческих ИТ-проектов и производить поиск проектов по текстовому запросу. Проведена оценка подходящего метода хранения файлов итоговых отчетов, выбран метод хранения. Разработана схема модерации проектов и реализована в виде функционального модуля сервиса. Произведена интеграция с git-сервисом Gitea для

хранения программного кода и взаимодействия через Api-интерфейс, настроено подключение к Authentication серверу Keycloak, используя технологию OpenID Connect.

В дальнейшем планируется рассмотреть создание системы анализа структуры итогового отчета с учетом заданных шаблонов для расширения функционала поисковой системы.

СПИСОК ЛИТЕРАТУРЫ

1. Боганюк Ю.В. Разработка системы для управления профессиональным развитием студента на основе его цифрового следа / Ю.В. Боганюк, М.С. Воробьева, И.Г. Захарова. — DOI: 10.15827/0236-235X.139.518-526 // Программные продукты и системы. 2022. — Т. 35. № 3. С. 518-526. — URL: https://cyberleninka.ru/article/n/razrabotka-sistemy-dlya-upravleniya-professionalnym-razvitiem-studenta-na-osnove-ego-tsifrovogo-sleda (дата обращения: 31.05.2023).
2. Зайцева С.А. Аксиологический подход к понятию цифрового следа / С.А. Зайцева, В.А. Смирнов. — DOI: 10.46724/NOOS.2021.3.79-87 // Ноосферные исследования. — 2012. — № 3. — С. 79-87. — URL: https://cyberleninka.ru/article/n/aksiologicheskiy-podhod-k-ponyatiyu-tsifrovogo-sleda (дата обращения: 31.05.2023).
3. Круглик Р. И. Обзор преимуществ и недостатков концепции MVC / Р. И. Круглик // Постулат. — 2019. — № 1-1(39). — С. 78. — URL: http://e-postulat.ru/index.php/Postulat/article/viewFile/2324/2364 (дата обращения: 31.05.2023).
4. Jaiswal P. Competitive analysis of web development frameworks / P. Jaiswal, S. Heliwal. — DOI: 10.1109/PAIS56586.2022.9946920. //Sustainable Communication Networks and Application: Proceedings of ICSCN 2021. — 2022. — P. 709-717. — URL: https://link.springer.com/chapter/10.1007/978-981-16-6605-6_53 (дата обращения: 31.05.2023).
5. de Almeida M. G. Authentication and authorization in microservices architecture: A systematic literature review / M. G. de Almeida, E. D. Canedo. //Applied Sciences. — 2022. — Т. 12, № 6. — С. 3023. — URL: https://www.mdpi.com/2076-3417/12/6/3023#text (дата обращения: 31.05.2023).
6. Sears R. To blob or not to blob: Large object storage in a database or a filesystem? / R. Sears, C. Van Ingen, J. Gray. — DOI: 10.48550/arXiv.cs/0701168. // Technical Report MSR-TR-2006-45 Microsoft Research Microsoft Corporation One Microsoft Way. — 2007. — URL:

https://www.microsoft.com/en-us/research/publication/to-blob-or-not-to-blob-large-object-storage-in-a-database-or-a-filesystem/ (date of the application: 31.05.2023)
7. Holzinger A. Applying Model-View-Controller (MVC) in design and development of information systems: An example of smart assistive script breakdown in an e-business application / A. Holzinger, K. H. Struggl, M. Debevc // International Conference on e-Business (ICE-B) 26-28 July 2010. Athens, 2010. — P. 1-6. — URL: https://graz.elsevierpure.com/en/publications/applying-model-view-controller-mvc-in-design-and-development-of-i#viewer-mode (дата обращения: 31.05.2023).