Общество с ограниченной ответственностью "ЦИФРОВАЯ ПЛАТФОРМА СОВМЕСТНОГО ИСПОЛЬЗОВАНИЯ АКТИВОВ"

УДК: 658.152 / 153
Регистрационный №122121300061-9
Инв. №1

УТВЕРЖДАЮ
Генеральный директор
Гостилович А. О.

ОТЧЕТ
о выполнении НИОКР по теме:
«Разработка и тестирование прототипа цифровой платформы совместного использования производственных активов»
(договор №4784ГС1/80287 от 18.11.2022)

Этап №1 «Разработка интерфейса прототипа цифровой платформы совместного использования активов
Разработка архитектуры прототипа цифровой платформы совместного использования активов
Программная реализация интерфейса прототипа цифровой платформы совместного использования активов
Разработка стандарта и спецификации для внедрения новых модулей системы
Разработка модуля авторизации для входа в личный кабинет
Разработка модуля взаимодействия с данными из ФНС
Формирование стандартных форм договоров аренды
Разработка модуля взаимной оценки и отзывов пользователей
Формализация методологии оценки индексов лояльности (NPS) и удовлетворенности (CSI) пользователей
Разработка серверной части прототипа цифровой платформы совместного использования активов
Программная реализация и тестирование серверной части прототипа цифровой платформы совместного использования активов»

(промежуточный)

Москва, 2023

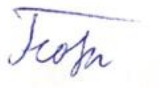
Руководитель работ  Гостилович А. О.

Москва, 2023

1. Реферат

Отчет 1 ч., 74 с., 1 формула, 0 рис., 2 табл., 59 источн., 7 прил.
цифровая платформа, аренда оборудования, совместное использование активов, недоиспользованное оборудование, управления загрузкой производственных фондов, интерфейс цифровой платформы, архитектура цифровой платформы, шеринг экономика
Объектом исследования является прототип цифровой платформы совместного использования активов
Целями текущего этапа НИОКР являются:
1) Формирование основных функциональных, технических и архитектурных требований к прототипу цифровой платформы совместного использования активов (ЦПСИА);
2) Разработка основных элементов ЦПСИА, а также модулей авторизации, взаимной оценки, взаимодействия с данными из ФНС и формирования стандартных форм договоров;
3) Программная реализация интерфейса и серверной части цифровой платформы.
В рамках текущего этапа НИОКР были сформулированы функциональные, технические и архитектурные требования к прототипу ЦПСИА, а также ценностное предложение для прототипа ЦПСИА: «Единый интернет-ресурс поиска заказов для загрузки простаивающих мощностей. Владельцы оборудования могут заработать на новых, бывших в употреблении и простаивающих мощностях, а арендаторы быстро найти производственные мощности у надежного поставщика по рыночным ценам для решения широкого спектра задач». На базе ценностного предложения были сформирован путь пользователя, на основе которого была проведена разработка основных элементов прототипа ЦПСИА, удовлетворяющих сформированным ранее требованиям, формализована методология оценки лояльности и удовлетворенности пользователей.
В процессе разработке требуемых элементов прототипа ЦПСИА, была проведана разработка интерфейса, архитектуры, стандарта и спецификации внедрения новых модулей, серверной части, модуля авторизации и входа в личный кабинет, модуля взаимодействия с данными из ФНС, модуля формирования стандартных форм договоров аренды и модуля взаимной оценки и отзывов пользователей. Далее была осуществлена программная реализация интерфейса и серверной части прототипа ЦПСИА, в процессе которой был выбран и объяснён стек технологий разработки, включающий современные, надёжные и производительные технологии. Также была протестирована серверная часть на корректность работы.
Таким образом было написано требуемое в рамках текущего этапа программное обеспечение в полном объёме, а также разработана требуемая документация: стандарт и спецификации внедрения новых модулей, полностью описанные в тексте отчёта. Выполнение указанных работ первого этапа служит основой для последующего развертывания прототипа ЦПСИА, доработки основных элементов системы и разработки необходимых модулей на втором этапе НИОКР.

Список исполнителей

Генеральный директор, кандидат экономических наук		А.О. Гостилович (введение, 1.5, 1.6, 2.1, 2.6, заключение)
Разработчик и архитектор системы, аспирант		С.О. Гостилович (1.1 – 1.4, 2.2 – 2.5, 2.7, 2.8, 3.1 – 3.2)
Научный руководитель, доктор экономических наук, профессор		Л.В. Лапидус (введение, 1.5, заключение)
Соисполнитель, Генеральный директор ООО «ФФФ Диджитал»		Д.Р. Хамитов (2.1, 3.1, 3.2)

Москва, 2023

Содержание

Москва, 2023

Глоссарий

Определения:

Авторизация – предоставление определенному лицу или группе лиц прав на выполнение определенных действий.

Агрегатор (или роутер) – модуль серверной части, отвечающий за маршрутизацию между запросами, поступающими от клиентской части и функциями, реализующими бизнес логику.

Аутентификация – проверка подлинности данных о пользователе.

Балансировщик нагрузки (Load Balancing) – это технология позволяющая распределять входящий трафик между несколькими используемыми серверами, тем самым, не допуская перегрузки одного из них.

Биллинговая система — программный комплекс, осуществляющий учет объема потребляемых абонентами услуг, расчет и списание денежных средств в соответствии с тарифами компании.

Веб-фреймворк — это программный фреймворк, предназначенный для поддержки разработки веб-приложений.

Виджет (widget) – слой в методологии FSD, представляющий собой самостоятельные блок интерфейса.

Клиентская часть (или Фронтэнд, от англ. Frontend) – это программное обеспечение, которое работает на компьютере пользователя (мобильном устройстве) и может взаимодействовать с сервером (серверной частью).

Линтер (от англ. слова lint) - статический анализатор кода, который указывает на возможно неправильные (подозрительные) участки программы, следовательно, помогая программисту писать более качественный код.

Микросервис – часть приложения, отвечающая за один элемент функционала, работающая в собственном процессе и коммуницирующая с остальными частями приложения.

Одностраничное веб-приложение (Single Page Application или SPA) – веб приложение или веб-сайт, использующий единственный HTML-документ как оболочку для всех веб-страниц и организующий

взаимодействие с пользователем через динамически подгружаемые HTML, CSS, JavaScript.

Прокси (Прокси-сервер) – это промежуточный сервер, выполняющий роль посредника между пользователем и целевым сервером, позволяющий клиентам как выполнять косвенные запросы (принимая и передавая их через прокси-сервер) к другим сетевым службам, так и получать ответы.

Серверная часть (или Бекэнд, от англ. Backend) – это программно-аппаратная часть сервиса, которая хранится на сервере, обрабатывает полученные данные и отправляет ответ обратно.

Стек технологий разработки (technology stack) – это комбинация программного обеспечения, языков программирования, фреймворков и технологий хранения данных, с помощью которых разработчики могут создавать и запускать компьютерные программы.

Токен или Токен Авторизации – эта специальный код (последовательность символов), хранящий в себе определённую информацию в закодированном виде и разрешающий доступ к данным конкретного пользователя.

Фича (от англ feature) – это переменная, которая описывает отдельную характеристику, свойство или признак объекта. Также это название используется методологии FSD, для обозначения элементов интерфейса, связанных с действиями, которые несут бизнес-ценность для пользователя.

Форматер кода – специализированная программа, осуществляющая автоматическое форматирование кода, обеспечивая соответствие заданным стандартам и правилам.

Фреймворк - программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Хеш – результат работы хеш-функции.

Хеш-функция (англ. hash function от hash) – функция, осуществляющая преобразование массива входных данных произвольной длины в строку установленной длины, выполняемое определённым алгоритмом.

Цифровая платформа – это основанная на программном обеспечении онлайн-инфраструктура, которая облегчает взаимодействие и транзакции между пользователями.

Шеринговая цифровая платформа (от англ. Sharing platform) - цифровая платформа, специализирующаяся на совместном использовании какого-либо ресурса, товара или услуги.

4G – это четвертое поколение технологии широкополосной сотовой связи.

API (Application Programming Interface) – описание способов взаимодействия одной компьютерной программы с другими.

CI/CD конвейер (continuous integration / continuous deployment) — это комбинация непрерывной интеграции (continuous integration) и непрерывного развертывания (continuous delivery) программного обеспечения в процессе разработки, объединяющее разработку, тестирование и развёртывание приложений.

CRM (CRM-система, от Customer relationship management) – это программное обеспечение для управления взаимоотношениями с клиентами.

CRUD (Creat, Read, Update, Delete) - акроним, обозначающий четыре базовые функции, используемые при работе с базами данных: создание (create), чтение (read), модификация (update) и удаление (delete).

CSI (Customer Satisfaction Index) – индекс потребительской удовлетворённости.

ERP (ERP-система, от Enterprise Resource Planning) – это программное обеспечение для управления бизнес-процессами и ресурсами предприятия.

FSD (Feature-Sliced Design) - архитектурная методология разработки клиентской части.

Git – широко распространённая система управления версиями исходного кода программ с распределенной архитектурой.

HTML – стандартизированный язык гипертекстовой разметки документов для просмотра веб-страниц в браузере.

HTTP — это протокол передачи гипертекстовой разметки, которая используется для передачи данных в интернете.

HTTPS – это защищенная версия HTTP протокола, использующая шифрование при передаче данных через HTTP протокол.

MVC (Model-View-Controller) – архитектурный паттерн (шаблон), при котором приложение разделяется на три отдельных компонента: модель (Model), представление (View) и контроллер (Controller).

NoSQL БД – нереляционная база данных.

NPS (Net Promoter Score) - индекс потребительской лояльности.

SOA (Service-Oriented Architecture) – сервис ориентированная архитектура.

SPA (Single Page Application) - одностраничное веб-приложение.

SQL (Structured Query Language) – язык структурированных запросов, использующийся для работы с реляционными базами данных.

SQL БД – реляционная база данных.

Unit-тесты — это программа, которая проверяет (тестирует) работу модуля или небольшой части кода.

URL (Uniform Resource Locator) - унифицированный указатель ресурса.

Сокращения:

БД – база данных.

НИОКР – научно-исследовательская и опытно-конструкторская работа.

ПО – программное обеспечение.

ФНС – федеральная налоговая служба.

ЦПСИА – цифровая платформа совместного использования активов.

Введение

Цель НИОКР: разработать и протестировать прототип цифровой платформы совместного использования производственных активов.

Выполнение НИОКР направлено на решение следующих научно-технических проблем: сокращение времени вынужденного простоя производственных мощностей с помощью цифровой платформы; повышение эффективности работы оборудования за счет внедрения новых решений для управления загрузкой производственных фондов.

В рамках текущего этапа НИОКР основными целями являются:

1) Формирование основных функциональных, технических и архитектурных требований к прототипу цифровой платформы совместного использования активов (ЦПСИА);

2) Разработка основных элементов ЦПСИА, а также модулей авторизации, взаимной оценки, взаимодействия с данными из ФНС и формирования стандартных форм договоров;

3) Программная реализация интерфейса и серверной части цифровой платформы.

Для выполнения данных целей необходимо выполнить следующие задачи (работы), указанные в календарном плане:

- разработка интерфейса прототипа ЦПСИА;
- разработка архитектуры ЦПСИА;
- разработка стандарта и спецификации для внедрения новых модулей системы;
- разработка модуля авторизации для входа в личный кабинет;
- разработка модуля взаимодействия с данными из ФНС;
- разработка модуля взаимной оценки и отзывов пользователей;
- разработка серверной части ЦПСИА;
- программная реализация интерфейса ЦПСИА;
- программная реализация и тестирование серверной части прототипа ЦПСИА;

- формирование стандартных форм договоров аренды;
- формализация методологии оценки индексов лояльности (NPS) и удовлетворенности (CSI) пользователей.

Выполнение указанных работ первого этапа служит основой для последующего развертывания прототипа ЦПСИА, доработки основных элементов системы и разработки необходимых модулей на втором этапе НИОКР.

Формирование требований к прототипу цифровой платформы совместного использования активов (ЦПСИА)

Основными целями, которые ставятся перед разрабатываемой цифровой платформой являются сокращение времени вынужденного простоя производственных мощностей с помощью цифровой платформы и повышение эффективности работы оборудования за счет внедрения новых решений для управления загрузкой производственных фондов. Также, необходимо обеспечить рыночную устойчивость и прибыльность разрабатываемой цифровой платформы. Дополнительным аспектом является то, что запуск и развитие цифровой платформы совместного использования активов необходимо проводить в несколько этапов, соответствующих жизненному циклу шеринговых цифровых платформ.

Таким образом, можно сформулировать требования к прототипу ЦПСИА, которые позволят осуществить решение поставленных целей и задач НИОКР. Основные требования можно разделить на функциональные, технические и архитектурные. Также необходимо проработать бизнес-модель разрабатываемой цифровой платформы. Отдельно, необходимо выделить особенность шеринговых платформ как цифровых сервисов, заключающуюся в необходимости оценки удовлетворённости и лояльности пользователей.

Функциональные требования к ЦПСИА

Прототип ЦПСИА должен реализовывать следующие функциональные возможности в рамках первого этапа НИОКР:

- предоставлять единую цифровую точку для взаимодействия участников;
- обеспечить централизованную идентификацию, аутентификацию и авторизацию пользователей с использованием единых справочников, реестров и классификаторов, в том числе внешних (данные Федеральной Налоговая Служба (ФНС) и т. п.);
- обеспечить инструменты взаимной оценки и работу системы отзывов и рекомендаций;

- обеспечить непрерывное совершенствование процессов на основе обратной связи от пользователей и оценки их уровня удовлетворенности и лояльности.

Технические требования к прототипу ЦПСИА

В качестве технических требований к разрабатываемому прототипу цифровой платформы можно выделить количественные параметры и структурные требования, включающие предполагаемый стек технологий разработки (набор языков программирования, библиотек и фреймворков).

В качестве количественных параметров можно выделить следующие:

- время загрузки первого контента: 0 – 6 сек.;
- индекс скорости загрузки: 0 – 9 сек. (в условиях 4G);
- время загрузки достаточной части контента: 0 – 7 сек. (в условиях 4G);
- время загрузки для взаимодействия: 0 – 8 сек. (в условиях 4G);
- максимальная потенциальная задержка после первого ввода: 0 – 1 сек.;
- количество возможных размещённых заявок: не менее 1000;
- доступность платформы: 24/7;
- выдерживаемая нагрузка на сайт: не менее 1000 посещений/день;
- синхронизации между базами данных цифровой платформы и информационными системами предприятий не более 2-х часов;
- использование 2-х видов электронно-цифровой подписи;
- время проведения 1 транзакции через платформу до 3 мин.;
- хранение информации о транзакциях не менее 6 мес.;
- помощью встроенного мессенджера (время обмена сообщениями от 0 – 6 сек.);
- количество одновременных пользователей чата не менее 100;
- использовать не менее 7 фичей для обучения модели машинного обучения рекомендательной системы;

- обновление исторических данных (данные логирования для визуализации, анализа и принятия решения) не менее 1 раза в 5 мин;
- период хранения исторических данных без сжатия не менее 3-х мес.

В качестве структурных требований можно выделить следующие:

- серверная часть должна включать базы данных (SQL, NoSQL);
- серверная часть должна реализовывать динамическое формирование страниц сайта;
- должна быть реализована возможность использования десктопной и мобильная версий сайта;
- языки разработки серверной части: PHP, Python, NodeJS (JavaScript);
- развернутая система должна включать в себя резервные каналы связи, балансировщик нагрузки (Load Balancing), системы для мониторинга и управления инфраструктурой.

Перечисленные требования легли в основу и ориентира выполнения работ первого этапа календарного плана НИОКР. Тестирование прототипа ЦПСИА на предмет соответствия перечисленным требования будет осуществлено в рамках второго этапа НИОКР.

Архитектурные требования к прототипу ЦПСИА

Архитектура разрабатываемого прототипа ЦПСИА должна обеспечивать следующие возможности:

- возможность масштабирования и долгосрочного ведения проекта;
- вертикальное и горизонтальное масштабирование (в плане функционала и количества пользователей);
- поддержка единых стандартов, правил поведения и инструментов для разработки сервисов прикладной бизнес-логики;
- архитектура системы, базирующаяся на принципах сервис-ориентированной архитектуры (SOA);

- адаптивность архитектуры под мобильные и десктопные устройства.

Подробное описание и обоснование выбранных архитектурных паттернов представлено на в пункте 2.2.

Бизнес-модель

Процесс создания прототипа ЦПСИА опирается на методологию бережливого стартапа (lean startup), раскрытую в трудах Стива Бланка и Боба Дорфа[1], а также Эрика Риса[2] и Александра Остервальдера и Ива Пинье[3].

Главной задачей на этапе создания прототипа является процесс развития потребителя (Consumer Development). Сама модель развития потребителей включает 4 этапа: выявление потребителей, верификация потребителей, расширение клиентской базы, выстраивание компании. Особенно актуален на этапе создания прототипа этап выявление потребителей, который включает в себя следующие шаги:

1. шаг формулировки гипотез — о продукте и потребителях, о дистрибуции и ценообразовании;
2. шаг тестирования и уточнения гипотез;
3. шаг тестирования и усовершенствования концепции продукта;
4. шаг верификации.

В рамках отчетного периода НИОКР были пройдены первые 3 этапа. Сформулированные гипотезы по сферам были проверены с помощью глубинных интервью с потенциальными пользователями (4 шт.), результаты представлены в таблице 1.1.

Таблица 1.1. Тестирование гипотез на этапе выявления потребителей

Тип гипотезы	Формулировка гипотезы	Результаты тестирования

[1] Бланк С., Дорф Б. Стартап: Настольная книга основателя / Стив Бланк, Боб Дорф; Пер. с англ. — М.: Альпина Паблишер, 2013. – 485 с. ISBN 978-5-9614-2809-4.

[2] Рис Э. Бизнес с нуля: Метод Lean Startup для быстрого тестирования идей и выбора бизнес-модели / Эрик Рис ; Пер. с англ. – 8-е изд. – М. : Альпина Паблишер, 2022. – 255 с. ISBN 978-5-9614-6837-3

[3] Остервальдер А., Пинье И. Построение бизнес-моделей: Настольная книга стратега и новатора. Пер. с англ. – М. : Альпина Диджитал, - 330 с. ISBN:9785961423457

Тип гипотезы	Формулировка гипотезы	Результаты тестирования
О продукте	Продукт должен представлять собой единую цифровую точку для взаимодействия участников (единый интернет-ресурс в глазах пользователей)	Гипотеза подтвердилась без уточнений
О потребителях	Потребителями ЦПСИА будут являться промышленные компании, которые продают бывшее в употреблении оборудование в Интернете	Гипотеза уточнилась: Потребители ЦПСИА делятся на 3 сегмента: компании, которые продают бывшее в употреблении оборудование в Интернете (1); компании, которые предоставляют в аренду спецтехнику с экипажем (2); компании, которые готовы изготавливать продукцию на заказ (3).
О дистрибьюции	Основным каналом привлечения пользователей будут классифайды, где размещены объявления о продаже бывшего в употреблении оборудования	Гипотеза уточнилась: Основными каналами привлечения пользователей будут: классифайды, тематические группы в телеграмм, тематические форумы.
О ценообразовании	Пользователи готовы платить 5% со сделки в пользу платформы за услуги финансового гаранта при оплате через сайт	Гипотеза подтвердилась для сегмента потребителей в сфере аренды спецтехники с экипажем, при этом для других сегментов модель ценообразования будет существенно отличаться. В сегменте аренды оборудования пользователи готовы платить комиссию на уровне 20%, но она должна включать в себя услуги по страхованию рисков. В сегменте контрактного производства компании не готовы платить комиссию.

Таким образом, была подтверждена гипотеза только о продукте, остальные гипотезы были уточнены. Главным изменением стала фокусировка на три сегмента потребителей (таблица 1.2).

Таблица 1.2. Характеристика сегментов пользователей прототипа ЦПСИА

Потребительские сегменты	Пользователи со стороны предложения	Пользователи со стороны спроса
Аренда спецтехники с экипажем	Прокатные компании, которые занимаются сдачей в аренду спецтехники с экипажем	Строительные компании, подрядчики строительных компаний
Аренда оборудования	Промышленные компании, у которых есть в наличии недоиспользованное или бывшее в употреблении оборудование	Промышленные компании, которые осваивают новые виды производства или новые компании, минимизирующие
Контрактное производство	Промышленные предприятия, у которых простаивают производственные линии	Крупные компании из различных сфер, которым интересно производство собственных торговых марок

Уточненные гипотезы на этапе выявления потребителей позволили сформировать следующее ценностное предложение для прототипа ЦПСИА: «Единый интернет-ресурс поиска заказов для загрузки простаивающих мощностей. Владельцы оборудования могут заработать на новых, бывших в употреблении и простаивающих мощностях, а арендаторы быстро найти производственные мощности у надежного поставщика по рыночным ценам для решения широкого спектра задач».

Формализация методологии оценки индексов лояльности (NPS) и удовлетворенности (CSI) пользователей

Для измерения лояльности пользователей к прототипу ЦПСИА используется индекс потребительской лояльности (Net Promoter Score, NPS). Данный индекс был предложен Фредериком Райхельдом в 2003 году[4] и с тех пор стал широко использоваться в практической деятельности компаний. Методология оценки индекса NPS предполагает обработку и анализ ответов пользователей сервиса на вопрос «Насколько вероятно, что Вы будете рекомендовать услуги компании своим друзьям, знакомым? (оцените по 10-

[4] Reichheld F. «The one number you need to grow» // Harvard Business Review, vol. 81, no. 12, pp. 46-54, 2003.

балльной шкале: от 1 балла (точно не буду рекомендовать) до 10 баллов (обязательно буду рекомендовать) в соответствии с Вашими ощущениями)».

В зависимости от ответов, пользователи платформы разделяются на три группы:

1. “Промоутеры” – самые лояльные потребители услуг, которые готовы рекомендовать услуги ЦПСИА своим знакомым (поставили 9 – 10 баллов);
2. “Нейтралы” – нейтральные (пассивные) потребители услуг, которых устраивает качество услуг ЦПСИА, но в целом они не лояльны к сервису (поставили 7 – 8 баллов);
3. “Критики” – потребители услуг, которые не удовлетворены услугами ЦПСИА и не лояльны к платформе (поставили 1 – 6 баллов).

Индекс NPS равен разнице между долей «Промоутеров» и «Критиков» в общем числе респондентов, оценивается в диапазоне от минус 100% до плюс 100%.

Наиболее популярной методикой оценки удовлетворённости пользователей различными сервисами является оценка CSI (Customer Satisfaction Index, CSI)[5]. Из разнообразия индексов, оценивающих удовлетворённость потребителей и опирающихся на разную методологию, сферу применения и другие специфические особенности[6], для измерения уровня удовлетворенности пользователей ЦПСИА был выбран индекс потребительской удовлетворенности.

Методология оценки CSI предполагает использование вспомогательных показателей «важности» (B) и «удовлетворённости» ($У$) по детерминантам качества услуг ЦПСИА, измеряемые на основе субъективного выбора пользователя по шкале от 1 до 5 баллов. В итоге рассчитываются

[5] Hill N., G. Brierley, and R. MacDougall. 2003. How to Measure Customer Satisfaction. Gower Publishing, Hampshire. pp. 12-16.

[6] Eboli L., Mazulla G. (2009). A new customer satisfaction index for assessing the quality of transit services // Journal of Public Transport, 12 (3): pp. 21-37. – p. 3

индексы потребительской удовлетворенности (CSI_i) по каждому критерию качества (i= [1, n]) и интегральный индекс потребительской удовлетворенности (CSI_I), определяемый по формуле (1).

$$CSI_I = \frac{\sum_{i=1}^{n} B_i y_i}{n} \qquad (1)$$

где:

CSI_I – интегральный индекс потребительской удовлетворенности;

B_i – важность i-ого критерия;

$У_i$ – удовлетворенность по i-ому критерию;

n – количество критериев.

Дальнейшим шагом в расчетах является оценка максимального и минимального значений CSI_I. Расчет максимального значения CSI_I производился при условии, что степень удовлетворенности по всем критериям равна 5, а для расчета минимального значения – степень удовлетворенности равна 1. Используются следующие интервалы и варианты удовлетворённости пользователей ЦПСИА:

- 85,0% - 100,0% - восхищенный пользователь;
- 70,0% - 85,0% - удовлетворенный пользователь;
- 40,0% - 70,0% - нейтральный пользователь;
- 20,0% - 40,0% - неудовлетворенный пользователь.

Соответствие CSI_I одному из перечисленных диапазонов характеризует удовлетворенность пользователей ЦПСИА.

Разработка основных элементов прототипа ЦПСИА

Для простоты изложения дальнейшего материала будет использоваться терминология, относящаяся к аренде оборудования, например, карточка оборудования, сделка по аренде оборудования, арендатор и владелец (оборудования). Однако, описанные элементы относятся и к другим сегментам работы прототипа ЦПСИА (Аренда спецтехники с экипажем, Контрактное производство, смотри таблицу 1.2), если это не обговорено отдельно. Так, например, для аренды спецтехники с экипажем, карточка оборудованиея будет означать карточка спецтехники с экипажем, а для контрактного производства – карточка производства.

Перед разработкой прототипа ЦПСИА был спроектирован путь пользователя (User Flow), чтобы выявить необходимые элементы интерфейса, клиентской и серверной частей разрабатываемой платформы. В данном случае под интерфейсом подразумевается необходимые структурные элементы взаимодействия пользователя с прототипом разрабатываемой цифровой платформы. Сформированный путь пользователя разрабатываемого прототипа цифровой платформы представлен в Приложении 1 (см. Приложение 1).

Разработка интерфейса прототипа ЦПСИА

Базируясь на основных элементах пути пользователя, представленных в Приложении 1, можно разделить интерфейс прототипа ЦПСИА на следующие основные элементы:

1. неизменяемые части основных страниц клиентской части прототипа цифровой платформы, включающие верхний и нижний колонтитулы (header и footer), и базовые страницы сайта: «как это работает», «арендовать», «сдать в аренду», «о нас», «контакты»;
2. интерфейс поиска оборудования, включающий, строку поиска и фильтры поиска;
3. окна регистрации (аутентификации);

4. интерфейс карточки оборудования;
5. личные кабинеты Арендатора и Владельца;
6. интерфейсы элементов жизненного цикла сделки по аренде оборудования:
   - отправка заявки на аренду;
   - коммуникация между владельцем и арендатором;
   - подписание договора аренды;
   - взаимная оценка пользователей и оценка сделки.
7. дополнительные информационные блоки, для размещения текстовой, графической и видео информации (информационные баннеры, видео инструкции и др.).

Следует отметить, что в приведённом выше списке фигурируют только те элементы интерфейса, которые были реализованы в рамках текущего этапа НИОКР. Во время следующего этапа будет произведено заполнение сайта платформы контентом, оформление дизайна, и более детальная проработка элементов интерфейса жизненного цикла сделки. Контент, дизайн и детальная проработка интерфейса являются аспектами, постоянно обновляющимися на основе результатов общения с потенциальными пользователями платформы, обратной связи от пользователей платформы при оформлении реальных сделок по аренде оборудования, поэтому целесообразно осуществить указанные работы в рамках развертывания и доработки прототипа ЦПСИА, что планируется во время следующего этапа НИОКР.

Разработка архитектуры прототипа ЦПСИА

Для разработки архитектуры прототипа ЦПСИА необходимо учесть основные требования, перечисленные в пункте 1.3, специфику бизнес ориентации платформы, а также особенности её разработки и развертывания. В частности, нужно учитывать направленность платформы на работы с юридическими лицами (и индивидуальными предпринимателями), и

разработку прототипа цифровой платформы в рамках ограниченных временных и денежных ресурсов, типичных для стартапов.

Разрабатываемый прототип цифровой платформы является интернет-сайтом, поэтому за архитектурную базу платформы использовалось традиционное разделение на клиентскую (Frontend) и серверную (Backend) части, которые взаимодействуют между собой через заданный протокол. Клиентская часть представляет собой интерфейс веб-сайта (или веб-интерфейс), с которой непосредственно взаимодействует пользователь. Серверная часть представляет собой сервер, в котором происходит работа с базами данных и другими элементами цифровой платформы. В качестве протокола обмена информацией между частями сайта был выбран самый распространённый протокол в сети интернет HTTP\HTTPS.

Архитектура клиентской части

На текущий момент существует три основных подхода к построению клиентской частей сайтов[7]:

- одностраничные веб-приложения (Single Page Application или SPA);
- серверный рендеринг;
- генератор статичных сайтов.

В сфере цифровых платформ преобладает использование одностраничных веб-приложений, так как такой подход обеспечивает наибольшую скорость работы, а также гибкость и простоту масштабируемости функционала[8]. Кроме того, применение SPA подхода позволяет разгрузить серверную часть от необходимости генерировать динамические страницы сайта, что, в свою очередь, позволяет унифицировать обмен между серверной и клиентской частями сайта. Иными

[7] Шарапов, С. Ф. Обзор способов разработки клиентских веб-приложений и преимущества использования генератора статичных сайтов / С. Ф. Шарапов // Информационные технологии и математическое моделирование (ИТММ-2021) : материалы XX Международной конференции имени А. Ф. Терпугова, Томск, 01–05 декабря 2021 года. – Томск: Национальный исследовательский Томский государственный университет, 2022. – С. 24-28. – EDN JUABOG.

[8] Федотова, К. В. Обзор вариантов использования REST в современной архитектуре web-приложений / К. В. Федотова, А. Хоук // Безопасность городской среды : Материалы V Международной научно-практической конференции, Омск, 21–23 ноября 2017 года / Под ред. Е.Ю. Тюменцевой. – Омск: Омский государственный технический университет, 2018. – С. 410-412. – EDN USFJGD.

словами, в дальнейшем уже готовую серверную часть можно будет связать с другим видом клиентской части, например, с мобильным приложением, с минимальными модификациями серверной части. Учитывая приведённые факторы, данный подход был выбран для разрабатываемого прототипа ЦПСИА.

Следует отметить необходимость адаптивности веб-интерфейса разрабатываемого прототипа под мобильные устройства. Несмотря на то, что основными пользователями разрабатываемого прототипа будут юридические лица и индивидуальные предприниматели, работники которых будут пользоваться платформой за рабочим местом, возможность использования ЦПСИА на мобильном устройстве будет удобно для работников во время поездок, например, во время приёма и передачи арендованного оборудования. SPA подход при разработке клиентской части сайта позволяет разрабатывать одностраничные веб-приложения способные автоматически адаптироваться к использованию на мобильных устройствах[9].

В качестве архитектурной методологии проектирования клиентской части разрабатываемого прототипа был выбран подход Feature-Sliced Design (FSD), сильно упрощающий изменение клиентской части, следствием чего является масштабируемость и возможность оперативного изменения интерфейса сайта[10]. Кроме того, учитывая необходимость в динамичности развития и создания новых версий, целесообразным является разработка одностраничного веб-приложения клиентской части прототипа ЦПСИА на базе существующих фреймворков (Angular, React и Vue)[11], что позволит значительно оптимизировать операционные ресурсы и ускорить время разработки.

---

[9] Мырзагалиев, А. М. Современные и адаптивные веб-приложения с поддержкой мобильных устройств / А. М. Мырзагалиев // Актуальные научные исследования в современном мире. – 2018. – № 1-1(33). – С. 25-30. – EDN YNLPYM.

[10] Документация FSD [Электронный ресурс]: Feature-Sliced Design. URL: https://feature-sliced.design/ru/docs (дата обращения: 27.04.2023).

[11] Ротарь В. Г., Ткачев М., Трофимова А. Е. Обзор актуальных фреймворков для разработки клиентской части современных веб-приложений //Молодежь и современные информационные технологии: сборник трудов XVIII Международной научно-практической конференции студентов, аспирантов и молодых учёных, 22-26 марта 2021 г., г. Томск. – Томский политехнический университет, 2021. – С. 170-171.

Архитектура серверной части

Одним из первоначальных аспектов при проектировании серверной части веб-сайта, которым является прототип ЦПСИА - это выбор архитектурного стиля взаимодействия сервера с клиентской частью (клиентскими частями). Было принято решение использовать REST (Representational State Transfer) API[12,13,14], которые обеспечивает масштабируемость, общность интерфейсов серверной части, независимое внедрение новых компонентов и эффективное использование промежуточных компонентов (прокси, балансировщики нагрузки и др.).

В качестве базового архитектурного паттерна (шаблона) взаимодействия с интерфейсом цифровой платформы был выбран Model-View-Controller (MVC), упрощающий модульную разработку и сопровождение программного продукта[15]. В нашем случае элементы Controller и Model располагаются в серверной части, когда элемент View относится к клиентской части разрабатываемого прототипа.

Говоря про базовую архитектуру серверной части разрабатываемого прототипа ЦПСИА, следует выделить сервис-ориентированную архитектуру (service-oriented architecture или SOA)[16], получившее широкое распространение в крупных шеринговых цифровых платформах[17]. Разделение всего функционала на набор переиспользуемых сервисов позволяет существенно упростить масштабируемость, увеличить надёжность и осуществлять независимую разработку элементов серверной части. Данные

[12] Li L. et al. Design patterns and extensibility of REST API for networking applications //IEEE Transactions on Network and Service Management. – 2016. – Т. 13. – №. 1. – С. 154-167.

[13] Федотова, К. В. Обзор вариантов использования REST в современной архитектуре web-приложений / К. В. Федотова, А. Хоук // Безопасность городской среды : Материалы V Международной научно-практической конференции, Омск, 21–23 ноября 2017 года / Под ред. Е.Ю. Тюменцевой. – Омск: Омский государственный технический университет, 2018. – С. 410-412. – EDN USFJGD.

[14] Jukan A. et al. Lab 4—Connecting Edge and Cloud Tools with REST HTTP //Network of Things Engineering (NoTE) Lab. – Cham : Springer International Publishing, 2023. – С. 111-122.

[15] Маннанов, А. А. Разработка MVC паттерна для повышения эффективности разработки веб-приложений / А. А. Маннанов // Информационные технологии. Проблемы и решения. – 2019. – № 4(9). – С. 123-129. – EDN HEGPGX.

[16] Perrey R., Lycett M. Service-oriented architecture //2003 Symposium on Applications and the Internet Workshops, 2003. Proceedings. – IEEE, 2003. – С. 116-119.

[17] Гостилович А. О. Трансформация бизнес-моделей промышленных предприятий под воздействием экономики совместного потребления: дисс. … к-та экон. наук. М.: Интеллектуальная Система Тематического Исследования НАукометрических данных, 2022. 209 с

преимущества полностью реализуются при разделении всей сервисной части на набор не зависимых друг от друга микросервисов[18]. Однако, применение микросервисной архитектуры не целесообразно на начальных стадиях разработки цифровых платформ, так как увеличивает сложность разработки, тестирования, развертывания серверной части [19,20].

Чтобы избежать перечисленные негативные эффекты микросервисной архитектуры было принято решение проектировать прототип на базе модульной архитектуры, построенной на основе SOA. Схема выбранной архитектуры ЦПСИА представлена (в Приложении 2). При такой архитектуре, серверная часть прототипа цифровой платформы будет состоять из модулей (вместо микросервисов), которые будут обособлены друг от друга, но иметь общую базу данных и возможность взаимодействия между собой непосредственно в коде, а не через API. В дальнейшем при достижении большого числа пользователей модули можно будет переформировать в микросервисы без глобального изменения бизнес-логики. Для обеспечения этого были разработаны стандарт и спецификация для внедрения новых модулей, в которых прописаны требования к структуре новых модулей и алгоритм преобразования модулей в микросервисы (см. пункт 2.3).

Также следует отметить, что для обеспечения широкой масштабируемости и возможности быстрого роста было принято решение использовать облачные инфраструктуры, предоставляемые российскими компаниями в качестве аппаратной части для сервера прототипа ЦПСИА. Рассматривались облачные решения таких платформ, как Yandex Cloud, VK Cloud, Cloud (SberCloud) и другие. При выборе облачного сервиса учитывались следующие факторы, влияющие на архитектуру:

---

[18] Кадыров, К. А. Микросервисная архитектура / К. А. Кадыров, А. М. Сафин // Фундаментальные и прикладные научные исследования: актуальные вопросы, достижения и инновации : сборник статей LII Международной научно-практической конференции : в 2 ч., Пенза, 15 января 2022 года. Том Часть 1. – Пенза: Наука и Просвещение (ИП Гуляев Г.Ю.), 2022. – С. 97-99. – EDN DWIUHS.

[19] Дейкун В. С. СРАВНИТЕЛЬНЫЙ АНАЛИЗ МИКРОСЕРВИСНОЙ И МОНОЛИТНОЙ АРХИТЕКТУР //Студенческая наука: актуальные вопросы, достижения и инновации. – 2022. – С. 47-49.

[20] Михневич А. В. Микросервисная архитектура для разработки программного обеспечения. – 2022.

- предоставление полного спектра требуемых вычислительных мощностей;
- возможность автомаштабирования и автоматической балансировки нагрузки;
- наличие возможности подключить необходимое ПО, такое как базы данных, прокси и кэширование.

Использование облачной инфраструктуры позволяет обеспечить быстрое развертывание системы без существенных вложений на приобретение, установку и настройку серверной аппаратуры.

Разработка стандарта и спецификации для внедрения новых модулей

Был сформирован внутренний стандарт, который включает в себя список правил, алгоритмов и спецификаций, обеспечивающих соответствие модульной архитектуры прототипа ЦПСИА принципам сервис-ориентированной архитектуры.

В сформированный стандарт и спецификацию входят:

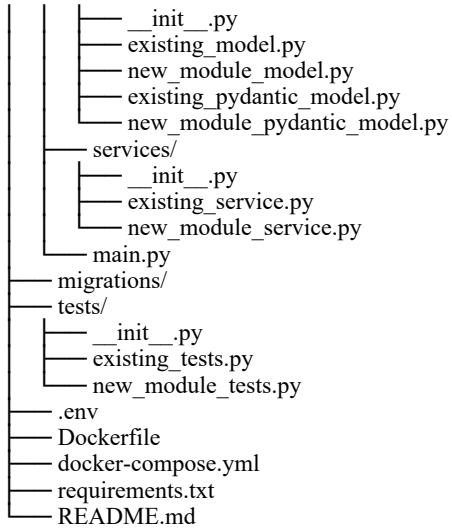
- спецификация файловой структуры прототипа ЦПСИА;
- принципы разработки нового модуля прототипа ЦПСИА;
- описание процесса разработки и внедрения нового модуля ЦПСИА;
- порядок преобразования модулей серверной части прототипа ЦПСИА в микросервисы.

Спецификация файловой структуры прототипа ЦПСИА

Спецификация файловой структуры при добавлении нового модуля прототипа ЦПСИА представляет собой изображение и описание условной файловой структуру серверной части разрабатываемого прототипа цифровой платформы и описание необходимых элементов нового модуля.

Файловая структура имеет следующий вид:

```
my_backend_app/
├── app/
│   ├── controllers/ (or routes/)
│   │   ├── __init__.py
│   │   ├── existing_controller.py
│   │   └── new_module_controller.py
│   ├── models/
```



При создание нового модуля (*new_module*) в файловую структуру необходимо добавить следующие элементы:

- файл контроллера *new_module_controller.py* в папке *controllers.* В этом файле прописывается маршруты перенаправление запросов от клиентской части разрабатываемого прототипа к функциям, реализующим функционал ответа на эти запросы, и управление взаимодействием между моделью и представлением;
- файлы моделей (сущностей в базе данных) *new_module_model.py* и *new_module_pydantic_model.py* в папке *models.* В этом файле определяются модели данных и схема базы данных для нового модуля. Также следует создавать *pydantic* модели, которые удобно использовать для автоматической валидации моделей данных;
- файл сервиса *new_module_service.py* в папке *services.* В этом файле реализуется функционал бизнес логики в виде переисползуемых функций (сервисов);
- файл тестов *new_module_tests.py* в папке *tests.* В этот файл должен содержать unit-тесты для функционала нового модуля. Покрытие модуля тестами должно быть не ниже 70%, при этом должны быть протестированы сценарии штатной работы, вызова исключений и граничные случаи, а также все возможные сценарии работы.

Кроме добавление указанных элементов, следует:

- обновить файлы __init__.py. Это позволит облегчить управление импортом и зависимостями во всем приложении;
- обновить main.py, включив в него новые маршруты из файла new_module_controller.py.

Принципы разработки нового модуля прототипа ЦПСИА

При разработке новых модулей прототипа ЦПСИА следует придерживаться следующих принципов, способствующих быстрому преобразованию модулей в микросервисы:

1. Реализовывать CRUD-операции на уровне сервисов. Это позволить обеспечить компактность контроллерной части модуля и её сфокусированность на обработке HTTP-запросов и ответов.

2. Избегать сильной связности модулей. Модули не должны использовать функции друг друга на уровнях: моделей и сервисов. Допускается (не желательно) взаимодействие на уровне контроллера.

3. Логическое разделение элементов баз данных, относящиеся к разным модулям. Например, в релятивистских базах данных необходимо выделять данные относящиеся к разным модулям в разные таблицы, что существенно упростит отделение базы данных этого модуля от общей базы данных при переходе к микросервисам.

Описание процесса разработки и внедрения нового модуля ЦПСИА

Любое изменение кода программы, включая разработку нового модуля, прототипа ЦПСИА следует осуществлять с использованием системы управления версиями на базе Git и в рамках сформированного CI/CD (Continuous Integration / Continuous Delivery) конвейера, обеспечивающего непрерывную интеграцию и развертывание модифицированного кода. Таким образом, можно выделить следующие требования при разработке нового модуля:

- разработку вести в отдельной ветке системы управления версиями на базе Git, отходящей от главной ветки разработки (development);

- использование систем контроля и повышения качества кода, таких как линтеров и форматеров;
- создание и проведение тестов разрабатываемого модуля, с покрытием не менее 70%, включая все возможные сценарии штатной и внештатной работы, вызовы исключений и граничные случаи;
- проведение тестов всей системы после добавления нового модуля;
- слияние ветки разработанного модуля в базовую ветку разработки (development) осуществлять через отправку запроса на слияния (pull request) и одобрение этого запроса ответственным лицом.

Для удовлетворения указанных выше требований необходимо и спецификации, описанной в пункте 2.3.1, необходимо придерживаться следующего алгоритма разработки и внедрения новых модулей:

1. В системе контроля версий создать новую ветку module_new_module выходящей из ветки development, в которой будет происходить разработка нового модуля.
2. Создать необходимые файлы в соответствующих папках, указанные в спецификации файловой структуры прототипа ЦПСИА (пункт 2.3.1):
   - *new_module_controller.py*
   - *new_module_model.py*
   - *new_module_pydantic_model.py*
   - *new_module_service.py*
   - *new_module_tests.py*
3. Обновить следующие файлы, указанные в спецификации файловой структуры прототипа ЦПСИА (пункт 2.3.1): *__init__.py*, *main.py.*
4. Провести разработку нового модуля, придерживаясь указанных выше принципов разработки нового модуля прототипа ЦПСИА (пункт 2.3.2)
5. Провести тестирование разработанного модуля, включая как тестирование нового модуля, так и тестирование всей системы после добавления нового модуля.

6. Осуществить проверку качества кода, с помощью систем контроля и качества кода, входящих в установленный CI/CD конвейер.

7. Выполнить запроса на слияния (pull request) в главную ветку разработки (development).

Порядок преобразования модулей серверной части прототипа ЦПСИА в микросервисы

Микросервис представляет собой полноценное приложение, запущенное изолированно от остальных частей приложения и взаимодействующее с ними по API через очередь сообщений.

Для преобразования модуля прототипа ЦПСИА в микросервис необходимо:

1. Создать отдельное приложение по структуре указанной в Спецификация файловой структуры прототипа ЦПСИА (пункт 2.3.1).

2. Переместить все файлы модуля в созданное приложение.

3. Создать собственные файлы main, db, config и тд. для управления работой сервиса.

4. Заменить все импорты из модулей на соответствующие функции в слое контроллера, которые будут обращаться к другим микросервисам или базовому приложению по API.

5. Заменить все импорты выделяемого в микросервис модуля в приложении на соответствующие вызовы по API.

6. Добавить файл Dockerfile для запуска микросервиса в отдельном контейнере.

7. Прописать логику запуска контейнера в файле Docker-compose.

Разработка модуля авторизации для входа в личный кабинет

Можно выделить три основных вида пользователей ЦПСИА: Гость (гостевой пользователь), Владелец и Арендатор. Гостевым пользователем является любой пользователь, который ещё не прошёл авторизацию. Этому типу пользователей будет предоставлена возможность изучения сайта прототипа ЦПСИА, включая использование определённой доли функционала

(ограниченный просмотр карточек оборудования, частичное создание собственной карточки оборудования и др.). Если пользователь хочет сдать оборудование в аренду, то он может пройти регистрацию как Владелец, а если пользователь хочет взять оборудование в аренду, то он может зарегистрироваться как Арендатор. Было принято решение разделить личные кабинеты Арендаторов и Владельцев, так как они имеют разный функционал и разный доступ к функциям ЦПСИА. Иными словами, если пользователь хочет быть и Владельцем, и Арендатором, то ему необходимо зарегистрировать 2 разных личных кабинета.

Кроме разделения на виды пользователей, в разрабатываемой цифровой платформе предполагается трёхуровневая иерархическая система доступа пользователей к функционалу платформы: Неавторизованный или Гостевой пользователь, авторизованный пользователь (Владелец или Арендатор) и активированный пользователь (Владелец или Арендатор). Гостевому (или неавторизованному) пользователю доступен ограниченный функционал, например, просмотр только части информации в карточках оборудования. Авторизованному пользователю доступен весь функционал гостевого пользователя и дополнительный функционал, например, просмотр полной информации в карточках оборудования). По аналогичному принципу активированный пользователь имеет возможность пользоваться всеми предоставленными функциями прототипа ЦПСИА, которые соответствуют его виду пользователя (Владелец или Арендатор), включая весь функционал предоставляющийся авторизованному пользователю. Блок схема иерархической структуры пользователей представлена в Приложении 3 (см. Приложение 3).

Следует отметить, что предложенная иерархия необходима для обсечения безопасного совершения сделки через разрабатываемую цифровую платформу. Для предотвращения возможность выставлять оборудование для аренды злоумышленниками, было принято решение осуществлять активацию зарегистрированного пользователя как Владельца.

Данная процедура будет проходить через предоставления дополнительной информации платформе и её проверку. Поскольку возможность мошенничества от Арендатора намного меньше, так как он передаёт деньги Владельцу через ЦПСИА, активация Арендатора до проведения сделки не требуется. Однако, было принято решение добавить активацию и Личного кабинета Арендатора, которая происходит при передаче денежных средств ЦПСИА в процессе заключения сделки. Таким образом, если Арендатор является активированным, то это означает, что у него уже были начаты сделки, в процессе которых он передавал деньги за аренду оборудования, что уменьшает вероятность его недобросовестности.

Что касается практической реализации модуля авторизации и входа в личный кабинет, то нужно выделить следующие два аспекта:

- реализация регистрации и последующей авторизации;
- реализация различного доступа к функционалу разрабатываемой платформы различными пользователями.

При регистрации пользователя в базе данных платформы не будет в прямом виде храниться пароль пользователя, а хеш этого пароля и дополнительные данные, чтобы обеспечить высокий уровень информационной безопасности.

Реализацию доступа к разному функционалу предлагается реализовать с применением JSON Web Token (JWT)[21], который является открытым стандартом RFC-7519[22] для создания токенов доступа. При использовании данного подхода во время авторизации серверная часть возвращает клиентской части специальный JWT токен, который клиентская часть отправляет вместе со следующими запросами к серверной части. Таким образом, при получении запроса, серверная часть может получить

[21] Ahmed S., Mahmood Q. An authentication based scheme for applications using JSON web token //2019 22nd international multitopic conference (INMIC). – IEEE, 2019. – С. 1-6.

[22] JSON Web Token, [Электронный ресурс]: RFC 7519. URL: https://www.rfc-editor.org/rfc/rfc7519 (дата обращения: 27.04.2023).

необходимую информацию для предоставления доступа к требуемому уровню функционала.

Разработка модуля взаимодействия с данными из ФНС

Основными клиентами разрабатываемой цифровой платформы будут являться юридический лица (и индивидуальные предприниматели), следовательно, при регистрации необходимо вводить данные о компании, которые будут использоваться при создании личного кабинета и карточек товара, формировании договоров, активации пользователя и в других процессах. В частности, необходимо будет указать следующие данные о регистрируемой компании: Краткое название, Полное название, ОГРН, ИНН, КПП, Дата регистрации, Уставный капитал, Юридический адрес, ФИО Руководителя, Основной вид деятельности.

Учитывая, что полный набор требуемых данных о любой российской компании содержатся в государственных реестрах (например, в реестрах Федеральной Налоговой Службы), было принято решение о добавления к прототипу ЦПСИА модуля взаимодействия с ФНС, которые позволяют осуществить автозаполнение всех необходимых данных. Иными словами, при заполнении информации, которая однозначно определяет юридическое лицо или индивидуального предпринимателя (например, ИНН), большинство остальных общедоступных данных будут загружены из реестра ФНС автоматически.

Таким образом, модуль взаимодействия с данными из ФНС позволит упростить процесс регистрации новых пользователей, а также уменьшить количество потенциальных ошибок при ручном вводе данных.

В рамках практической реализации, была выбран сервис DaData[23] предоставляющее программное API для работы с открытыми данными из ФНС. Выбор данного решения обусловлен большим количеством готовых сервисов работы с данными, которые в дальнейшем можно будет

[23] Сервис DaData [Электронный ресурс]: DaData. URL: https://dadata.ru/ (дата обращения: 02.05.2023).

унифицировано интегрировать в модуль взаимодействия с данными из ФНС. Согласно указанным тарифам, бесплатная версия сервиса включает 10000 запросов в сутки и всю необходимую часть информации, которая будет использоваться в прототипе ЦПСИА. В будущем возможен переход на платные тарифы при достижении высокого уровня трафика у разрабатываемой цифровой платформы. Также следует отметить, что данный сервис является российским проектом участника Инновационного центра Сколково, предоставляющий открытый исходный код, что устраняет риски прекращения доступа к сервису вследствие санкционных ограничений.

Формирование стандартных форм договоров аренды

Прототип цифровой платформы совместного использования активов должен включать в себя возможность формирования стандартных форм договоров аренды. Сформированные формы договоров различаются по виду аренды:

- договор аренды оборудования;
- договор аренды спецтехники с экипажем;
- договор на изготовление продукции.

На основе заполненных данных Личного кабинета пользователя в стандартных формах договоров аренды будет добавлена следующая информация:

- город и дата договора;
- название юридических лиц арендатора и арендодателя;
- сроки аренды;
- стоимость аренды;
- адреса и банковские реквизиты сторон.

Наличие функции формирования стандартных форм договоров аренды позволит упростить документооборот между пользователями платформы, снизить время на заключение сделки и юридические риски как для контрагентов, так и для оператора платформы.

Разработка модуля взаимной оценки и отзывов пользователей

Получение обратной связи от пользователей цифровой платформы является ключевым элементом адаптации платформы к клиентам и изменяющимся рыночным условиям[24]. Для шеренговых цифровых платформ также характерны взаимная оценка пользователей, что позволяет составить их рейтинг[25]. Рейтинг, в свою очередь, может являться частью бизнес-процесса и служить основой при проектировании рекомендательной системы.

В качестве базовой системы оценок пользователя была выбрана распространённая 5 бальная система, когда пользователь указывает количество звёзд (от 1 до 5) и имеет возможность оставить комментарий. Эта система будет применяться после окончания сделки для оценки другого пользователя, а также для получения обратной связи от пользователей.

Что касается технической реализации модуля оценки пользователей, в основной базе данных серверной части прототипа ЦПСИА будет сформирована таблица, включающая следующую информацию: ссылки на сделку, оценки сторон сделки, оценку платформы и комментарии (если имеются). Эта таблица в последствии будет использована в качестве источника параметров (фичей) для рекомендательной системы, архитектура которой будет разработана во время следующего этапа текущего НИОКР.

Разработка серверной части прототипа ЦПСИА

В процессе разработки серверной части прототипа ЦПСИА были проработаны аппаратное и инфраструктурное обеспечение сервера, базы данных, способ коммуникации с клиентской частью и структура серверного приложения. Под серверным приложением подразумевается основной

[24] Damschroder L. J. et al. The updated Consolidated Framework for Implementation Research based on user feedback //Implementation Science. – 2022. – Т. 17. – №. 1. – С. 1-16.

[25] Da Silveira A. B., Hoppen N., De Camillis P. K. Flattening relations in the sharing economy: A framework to analyze users, digital platforms, and providers //Handbook of Research on the Platform Economy and the Evolution of E-Commerce. – IGI Global, 2022. – С. 26-51.

элемент серверной части прототипа цифровой платформы, выполняющий весь функционал бизнес-логики.

Аппаратное и инфраструктурное обеспечение серверной части

Как было указано в пункте 2.2, в качестве аппаратной части сервера разрабатываемого прототипа цифровой платформы предложено использовать облачную инфраструктуру. В процессе разработки было принято решение использовать платформу Yandex Cloud (Яндекс Облако)[26], так как она удовлетворяет указанным в пункте 2.2.2 архитектурным требованиям и имеет широкий функционал[27]. Следует отметить, что вся инфраструктура Яндекс Облака защищена в соответствии с Федеральным законом Российской Федерации «О персональных данных» № 152-ФЗ, что позволяет обеспечить высокий уровень информационной безопасности. Кроме того, Яндекс Облако является крупной отечественной компанией, что сокращает риски ограничения доступа к её инфраструктуре вследствие внешних политических причин.

Яндекс Облако предоставляет разные возможности для использования своих ресурсов в качестве аппаратной части сервера, среди которых следует отметить следующие сервисы:

1. Yandex Compute Cloud

Сервис Yandex Compute Cloud предоставляет масштабируемые вычислительные мощности для создания виртуальных машин и управления ими. Сервис поддерживает прерываемые виртуальные машины, а также отказоустойчивые группы виртуальных машин. Существует возможность подключать к виртуальным машинам диски с образами на базе ОС Linux. Каждый диск автоматически реплицируется внутри своей зоны доступности, что обеспечивает надежное хранение данных. Также, для удобного переноса

[26] Логвинов, Д. В. Возможности отечественной облачной инфраструктуры на примере сервиса Yandex Cloud / Д. В. Логвинов, С. С. Савкин // Актуальные проблемы науки и образования в условиях современных вызовов : Сборник материалов XIII Международной научно-практической конференции, Москва, 15 августа 2022 года. – Москва: Печатный цех, 2022. – С. 23-25. – EDN DMKKOK.

[27] Управляемые сервисы в облаке [Электронный ресурс]: Yandex Cloud. URL: https://cloud.yandex.ru/services (дата обращения: 03.05.2023).

данных с одного диска на другой Compute Cloud поддерживает снимки дисков. Кроме того, Yandex Compute Cloud позволяет создавать виртуальные машины как из готовых образов, например, операционных систем, так и из собственных образов контейнеров, хранящихся в Yandex Container Registry.

2. Yandex Serverless Containers

Сервис Yandex Serverless Containers позволяет запускать контейнеризированные приложения в безопасном, отказоустойчивом и масштабируемом окружении без создания и обслуживания виртуальных машин.

3. Yandex Network Load Balancer

Сервис Yandex Network Load Balancer обеспечивает отказоустойчивость приложений за счет равномерного распределения сетевой нагрузки по облачным ресурсам. Сервис позволяет создавать сетевые балансировщики и объединять облачные ресурсы в целевые группы, по которым нужно распределять трафик. Ресурсы из подключенной к балансировщику целевой группы регулярно опрашиваются проверками состояния, благодаря чему трафик подается только на рабочие ресурсы.

4. Yandex Managed Service for PostgreSQL

Сервис Managed Service for PostgreSQL помогает разворачивать и поддерживать кластеры серверов PostgreSQL в инфраструктуре Yandex Cloud.

5. Yandex Message Queue

Yandex Message Queue — универсальное масштабируемое решение для обмена сообщениями между приложениями. Для работы с сервисом можно использовать популярные инструменты. Message Queue помогает:

- выстроить коммуникацию между отдельными приложениями системы;
- масштабировать систему, которая полагается на обмен информацией между отдельными приложениями;
- повысить отказоустойчивость обмена информацией при сбоях отдельных приложений;

- освободить ресурсы для обработки срочных запросов, перекладывая обработку входящих сообщений на предназначенные для этого приложения.

6. Yandex Cloud CDN

Yandex Cloud CDN предоставляет инструменты организации доставки контента до конечных потребителей с помощью сети распространения контента (Content Delivery Network, CDN), что, в свою очередь, позволяет:

- перенести часть нагрузки с источников данных на CDN-серверы за счет кеширования;
- сократить время доставки контента конечным пользователям.

Таким образом, при использовании Яндекс Облака в качестве аппаратного части сервера прототипа ЦПСИА имеется возможность запустить виртуальную машину (Yandex Compute Cloud) содержащую серверное приложение и необходимые базы данных. К виртуальной машине имеется возможность без изменения кода серверного приложения добавить балансировку нагрузки (Yandex Network Load Balancer) и распределённую сеть передачи контента (Yandex Cloud CDN), что позволит значительно улучшить производительность и надёжность сервера. При увеличении масштаба будет возможность использовать серверное решение работы с базой данных (Yandex Managed Service for PostgreSQL). В дальнейшем, при переходе от модульной архитектуры на базе принципов SOA к микросервисам можно будет, если будет целесообразно, вместо, виртуальной машины использовать бессерверное окружение Yandex Serverless Containers совместно с очередью сообщений Yandex Message Queue. Следует упомянуть, что использование Яндекс Облака будет осуществлено при развёртывании системы, которое запланировано в рамках следующего этапа НИОКР.

Говоря про инфраструктурное обеспечения серверной части прототипа ЦПСИА, можно выделить оставшиеся программные элементы сервера помимо непосредственно серверного приложения. К ним относятся базы данных, кэширование и другие элементы. В рамках текущего этапа НИОКР

этими элементами являются базы данных (подробнее описано ниже). Важным аспектом является необходимость контейнеризации всех элементов серверной части, включая серверное приложение и базы данных. Предполагается использовать контейнеризацию на базе Docker, что облегчает процесс развертывания и масштабирования[28].

Базы данных

Основная бизнес-логика прототипа ЦПСИА заключается в проведении сделок по аренде оборудования, что означает работу с финансовыми операциями, налагающими высокие требования к надежности и целостности базы данных. Для таких задач используются реляционные базы данных (БД) (или SQL БД)[29]. Кроме того, на сервере будут храниться данные, например, текстовые и медиа файлы, которые целесообразно помещать в (нереляционные) базы данных (noSQL БД)[30]. Таким образом в серверной части прототипа ЦПСИА будут использоваться SQL база данных для основной бизнес-логики и noSQL база данных для хранения файлов.

В реляционной базе данных были выделены следующие таблицы, выражающие основные сущности бизнес-логики:

- Пользователь. Вся информация о пользователе, требуемая для авторизации и автозаполнения форм и договоров;
- Компания. Таблица, содержащая информацию о компании (ИП, ООО и т.д.). Каждый пользователь должен относиться к компании;
- Оборудование. Таблица, содержащая основные поля, необходимые для описания объектов аренды;
- Заказы. Информация о заказах по аренде оборудования, включая статус заказа, который может означать заключение сделки;

[28] Закирова, Ю. М. Метод контейнеризации веб-приложений / Ю. М. Закирова // Приоритетные направления развития науки в современном мире : Сборник научных статей по материалам V Международной научно-практической конференции, Уфа, 05 февраля 2021 года. – Уфа: Общество с ограниченной ответственностью "Научно-издательский центр "Вестник науки", 2021. – С. 37-42. – EDN GXIXUO.

[29] Кириллов В. В. Введение в реляционные базы данных. – БХВ-Петербург, 2012.

[30] Мартишин С. А., Симонов В. Л., Храпченко М. В. Базы данных. Практическое применение СУБД SQL и NoSQL-типа для проектирования информационных систем. – 2017.

- Договоры. Документы, сформированные системой или загруженные извне, относящиеся к сделкам (заказам), проводимым на платформе;
- Оценки. Таблица оценок, выставляемых пользователями друг другу.

Данный набор таблиц обеспечивают возможность осуществить требуемый в рамках текущего этапа НИОКР функционал прототипа ЦПСИА.

Коммуникация клиент-сервер

Как было описано в пункте 2.2.2, в качестве архитектурного стиля (механизма) взаимодействия сервера с клиентской частью был выбран REST API, основанный на передаче HTTP-запросов и ответов. REST API определяет набор ограничений и соглашений для обмена данными между клиентом (клиентской частью) и сервером (серверной частью).

Процесс коммуникации с REST API включает следующие шаги:

1. Клиент отправляет HTTP-запрос на определенный URL (endpoint) сервера с использованием одного из методов (GET, POST, PUT, DELETE и т. д.).
2. Сервер обрабатывает запрос, выполняет соответствующие действия (например, чтение, создание, обновление или удаление данных) и возвращает HTTP-ответ.
3. Клиент обрабатывает ответ, преобразует данные в нужный формат и обновляет пользовательский интерфейс.

К основным аспектам, определившим выбор этого механизма взаимодействия между клиентской и серверной частей прототипа ЦПСИА, можно отнести следующие преимуществам использования REST API:

- Универсальность. REST API основан на стандартных принципах HTTP, что позволяет легко интегрироваться с любым клиентом, независимо от языка программирования или платформы;
- Безопасность. Используя HTTPS (защищённую версию HTTP протокола), можно обеспечить безопасность передачи данных между клиентом и сервером.

- Масштабируемость. REST API поддерживает кеширование, что может улучшить производительность и снизить нагрузку на сервер.
- Гибкость: REST позволяет разрабатывать API с множеством различных форматов данных (JSON, XML и т. д.).
- Самодокументирование. REST API является понятным и доступным для разработчиков благодаря структуре URL и использованию стандартных HTTP-методов.

Иными словами, использование REST API для коммуникации между клиентом и сервером, позволяет получить гибкую, масштабируемую и безопасную систему для передачи данных и реализации бизнес-логики. Применение принципов REST API обеспечивает простоту интеграции и возможность разрабатывать веб-приложения с высокой производительностью и удобством поддержки.

Структура серверного приложения

При разработке архитектуры серверной части прототипа ЦПСИА (пункт 2.2.2) в качестве базового архитектурного шаблона был выбран Model-View-Controller. MVC способствует четкому разделению обязанностей путем разделения приложения на три компонента: модель (Model), представление (View) и контроллер (Controller). Модель представляет данные и бизнес-логику приложения. Представление является репрезентацией слоя пользовательского интерфейса. Контроллер обрабатывает пользовательский ввод и управляет взаимодействием между моделью и представлением.

Следует упомянуть, что в текущей реализации прототипа ЦПСИА элемент представления реализован внешним образом и по сути является клиентской частью. Иными словами, серверная часть обрабатывает запросы клиентской части и возвращает данные в структурированном формате. Клиентская часть использует эти данные при отображении пользовательского интерфейса. Это разделение обязанностей позволяет достичь большей гибкости при разработке и поддержке как клиентской, так и серверной частей прототипа ЦПСИА.

В рамках текущей разработки серверной части прототипа ЦПСИА, компоненты архитектурного паттерна MVC представляют собой следующие элементы:

- Модель представляет данные и бизнес-логику приложения. Она отвечает за получение, хранение и обновление данных, а также инкапсуляцию правил и процессов. В серверной части разрабатываемого прототипа слой модели включает все модули серверного приложения, отвечающие за бизнес-логику и работу с базами данных.
- Контроллер отвечает за обработку пользовательского ввода, обработку запросов и управление взаимодействием между моделью и представлением. В приложении, ориентированном на REST API, контроллер в основном связан с обработкой входящих HTTP-запросов, вызовом соответствующей бизнес-логики и возвращением соответствующих HTTP-ответов. Иными словами, контроллером является агрегатор (или роутер), который осуществляет маршрутизацию между запросам (ответами) и вызовом нужных функций.
- Представление отвечает за отображение пользовательского интерфейса и представление данных. Этот слой полностью вынесен в клиентскую часть и обменивается информацией с сервером через HTTP протокол на базе REST API стиля клиент-серверного взаимодействия.

Согласно принципам модульной архитектуры на базе SOA слой модель, отвечающий за бизнес-логику, следует разделить на модули, обособленные друг от друга и выполняющие определённую часть функционала. Весь функционал реализующийся в рамках текущего этапа НИОКР был разделён на следующие модули:

- модуль работы с пользователями;
- модуль работы с оборудованием;
- модуль авторизации и входа в личный кабинет;

- модуль взаимодействия с данными из ФНС;
- модуль работы с договорами (включает формирование стандартных форм договора)
- модуль взаимной оценки и отзывов пользователей.

Структура серверной части прототипа ЦПСИА представлена в Приложении 4 (см. Приложение 4).

Реализация интерфейса и серверной части ЦПСИА

Программная реализация интерфейса прототипа ЦПСИА

Стек технологий разработки

1. React

React[31] — это популярный JavaScript-фреймворк для построения пользовательских интерфейсов. Он использует компонентный подход, что позволяет разработчикам создавать масштабируемые и легко поддерживаемые приложения. React обеспечивает высокую производительность благодаря использованию виртуального DOM-дерева и оптимизации рендеринга компонентов. Современные версии React предлагают функциональный подход к созданию компонент и множество вспомогательной функциональности - хуки. Это самый популярный на данный момент JavaScript-фреймворк, что обеспечивает высокую надежность, активную поддержку кодовой базы и обилие информации от сообщества.

2. React Router

React Router[32] - это мощная библиотека маршрутизации для React-приложений. Она позволяет создавать многопользовательские приложения с навигацией и переходами между страницами, сохраняя при этом единое кодовое дерево и оптимизируя загрузку данных. React Router также облегчает обработку URL-параметров и состояний перехода.

3. Redux

Redux[33] - это предсказуемый контейнер состояния для JavaScript-приложений. Он позволяет управлять состоянием приложения в едином хранилище, что облегчает отладку, тестирование и воспроизводимость ошибок. Redux интегрируется с React и предоставляет удобный API для

---

[31] Getting Started [Электронный ресурс]: React documentation. URL: https://reactjs.org/docs/getting-started.html (дата обращения: 03.05.2023).

[32] Feature Overview [Электронный ресурс]: React Router documentation. URL: https://reactrouter.com/en/6.11.1/start/overview (дата обращения: 03.05.2023).

[33] Getting Started with Redux [Электронный ресурс]: Redux documentation. URL: https://redux.js.org/introduction/getting-started (дата обращения: 03.05.2023).

обновления состояния компонентов на основе действий и изменений состояния приложения.

4. Axios

Axios - это популярная библиотека для выполнения HTTP-запросов в браузере и Node.js[34]. Она предоставляет простой и гибкий API для выполнения запросов, обработки ответов и ошибок. Axios также поддерживает промисы и async/await, что упрощает асинхронную работу с запросами и ответами сервера.

5. Material-UI

Material-UI - это исчерпывающая библиотека компонентов пользовательского интерфейса, основанная на принципах Material Design[35]. Она предлагает множество готовых компонентов и стилей, что ускоряет разработку и обеспечивает согласованный стиль дизайна. Material-UI также поддерживает кастомизацию тем и стилей, что позволяет легко привести компоненты в соответствие с вашим брендом и дизайн-гайдлайнами.

6. Formik

Formik - это библиотека для управления формами и валидации ввода данных в React-приложениях[36]. Она предоставляет удобный API и интеграцию с множеством библиотек валидации, что облегчает обработку пользовательского ввода и проверку корректности данных. Formik также поддерживает управление состоянием формы и обработку ошибок.

7. JSONWebToken

Jsonwebtoken (или JWT) - это библиотека для работы с JSON Web Tokens[37]. JWT - это стандарт для безопасной передачи информации между клиентом и сервером. В контексте аутентификации JWT используется для

---

[34] Axios [Электронный ресурс]: GitHub. URL: https://github.com/axios/axios (дата обращения: 03.05.2023).

[35] Usage [Электронный ресурс]: Material-UI documentation. URL: https://mui.com/getting-started/usage (дата обращения: 03.05.2023).

[36] Overview [Электронный ресурс]: Formik docs. URL: https://formik.org/docs/overview (дата обращения: 03.05.2023).

[37] Jsonwebtoken [Электронный ресурс]: GitHub. URL: https://github.com/auth0/node-jsonwebtoken (дата обращения: 03.05.2023).

создания, верификации и обновления токенов доступа и обеспечения безопасности пользовательских сессий.

8. Vite

Vite - это инструмент для сборки и разработки современных JavaScript-приложений[38]. Он предлагает быструю и легкую сборку, оптимизацию и инкрементальное обновление кода. Vite также поддерживает обработку CSS, изображений и других ресурсов, а интеграция с популярными фреймворками, такими как React, делает его удобным инструментом для разработки и деплоя проектов.

9. Sass

Sass (Syntactically Awesome StyleSheets) - это расширение CSS, которое добавляет дополнительные возможности и «синтаксический сахар» для упрощения и ускорения написания стилей[39]. Sass предлагает такие функции, как переменные, вложенные правила, миксины и функции, которые упрощают разработку и поддержку CSS-кода. Sass компилируется в обычный CSS, что делает его совместимым со всеми браузерами и инструментами разработки. Интеграция Sass с вашим сборщиком, таким как Vite, позволяет автоматически компилировать и оптимизировать стили во время разработки и сборки проекта.

10. ESLint

ESLint - это линтер для JavaScript, который помогает обеспечить согласованность стиля кода и предотвратить распространенные ошибки[40]. ESLint предлагает множество настраиваемых правил и плагинов для различных стилей кодирования и фреймворков, включая React. Интеграция ESLint со средой разработки позволяет автоматически проверять и исправлять код на этапе написания.

---

[38] Getting Started [Электронный ресурс]: Vite documentation. URL: https://vitejs.dev/guide/ (дата обращения: 03.05.2023).

[39] Documentation [Электронный ресурс]: Sass. URL: https://sass-lang.com/documentation (дата обращения: 03.05.2023).

[40] Getting Started with ESLint [Электронный ресурс]: ESLint documentation. URL: https://eslint.org/docs/user-guide/getting-started (дата обращения: 03.05.2023).

Этот стек разработки является оптимальным для сложных одностраничных веб-приложения на React по следующим причинам:

1. Современные технологии. Все предложенные инструменты и библиотеки являются современными и активно поддерживаются, что обеспечивает безопасность, производительность и стабильность приложения. Иными словами, выбранный стек технологий разработки должен обеспечить технические требования, указанные в пункте 1.2.
2. Широкое сообщество. Большинство инструментов в этом стеке имеют активное и большое сообщество, что облегчает поиск решений, примеров и документации для различных аспектов разработки.
3. Масштабируемость. Комбинация React, Redux, и Feature-Sliced Design позволяет разработчикам создавать масштабируемые приложения с хорошей организацией кода и легким внедрением новых функций.
4. Удобство разработки. Благодаря инструментам, таким как Vite и ESLint, разработка становится более удобной и быстрой, что ускоряет время разработки и упрощает поддержку кода.
5. Повышенная производительность. Использование таких инструментов, как React Router и axios, позволяет оптимизировать загрузку данных и переходы между страницами, что обеспечивает лучшую производительность приложения.
6. Гибкость. Предложенный стек дает разработчикам гибкость в выборе дополнительных инструментов и библиотек, чтобы лучше соответствовать требованиям проекта и предпочтениям команды.
7. Безопасность и аутентификация. Использование jsonwebtoken для работы с аутентификацией обеспечивает безопасность пользовательских сессий и защищает данные от несанкционированного доступа.

8. Улучшенный дизайн и стилизация. Благодаря использованию Material-UI и Sass разработчики могут быстро создавать красивые и согласованные пользовательские интерфейсы с возможностью легкой кастомизации и адаптации.

Реализация элементов интерфейса

Как описано в пункте 2.2.1, разработка клиентской части, входящая в программную реализацию интерфейса, основана на применении Feature-Sliced Design в качестве архитектурной методологии проектирования. FSD предлагает разделение кода на слои, сегменты и срезы, что делает код более понятным и упорядоченным. Эта методология, также, способствует лучшему взаимодействию команды разработки, упрощает навигацию по коду и делает приложение более гибким для изменений.

В рамках FSD программная реализация осуществляется разделение интерфейса одностраничного веб-приложения на следующие слои: Приложение (App), Процессы (Processes), Страницы (Pages), Виджеты (Widgets), Фичи (Feature), Сущности (Entities) и Совместные элементы (Shared). Основные элементы интерфейса прототипа ЦПСИА, указанные в пункте 2.1, относятся к уровню виджетов и, в редких случаях, к страницам.

В рамках текущего этапа НИОКР были программно реализованы все элементы интерфейса прототипа ЦПСИА, указанные в пункте пункте 2.1. В качестве иллюстрации, в приложении представлены следующие интерфейсы: Базовая страница клиентской части (Приложение 5., Рис.1., Рис.2.); интерфейс поиска оборудования (Приложение 5., Рис.3.); Окно регистрации нового пользователя (Приложение 5., Рис.4); Карточка оборудования (Приложение 5., Рис.5.); Личный кабинет Владельца (Приложение 5., Рис. 6); Интерфейс взаимной оценки пользователей и сделки (Приложение 5., Рис. 7). Следует отметить, что вся контактная информация (номера телефонов, название фирм и т.д.) были придуманы для примера.

Программная реализация и тестирование серверной части прототипа ЦПСИА

Основной стек технологий разработки

При программной реализации серверной части прототипа ЦПСИА были применены следующие технологии разработки:

1. Python 3.11

Python - интерпретируемый, высокоуровневый язык программирования общего назначения, известный своей читабельностью, простотой и обширной экосистемой библиотек и пакетов. Гибкость Python делает его подходящим языком для веб-разработки и широко используемым в отрасли[41]. Версия 3.11 содержит несколько новых функций и оптимизаций, которые повышают производительность и удобство использования[42].

2. FastAPI

FastAPI - это современный и высокопроизводительный веб-фреймворк для создания API с Python 3.7+ на основе стандартных подсказок типов Python. Он предлагает автоматическую проверку данных запроса и ответа, автоматическую генерацию интерактивной документации API, инъекцию зависимостей и другие возможности[43]. По производительности FastAPI находится на одном уровне с Node.js и Go, что делает его отличным выбором для создания REST API[44].

3. Uvicorn

Uvicorn - это ASGI (Asynchronous Server Gateway Interface) сервер, который обеспечивает больший параллелизм и способен обрабатывать несколько запросов одновременно[45]. Это рекомендуемый сервер для FastAPI благодаря его производительности, надежности и простоты интеграции.

---

[41] 8 World-Class Software Companies That Use Python, [Электронный ресурс]: Real Python. URL: https://realpython.com/world-class-companies-using-python/ (дата обращения: 02.05.2023).

[42] What's New In Python 3.11, [Электронный ресурс]: Python Documentation. URL: https://docs.python.org/3/whatsnew/3.11.html (дата обращения: 02.05.2023).

[43] FastAPI [Электронный ресурс]: fastapi.tiangolo URL: https://fastapi.tiangolo.com/ (дата обращения: 02.05.2023).

[44] Web Framework Benchmarks [Электронный ресурс]: Techempower. URL: https://www.techempower.com/benchmarks/ (дата обращения: 02.05.2023).

[45] Uvicorn [Электронный ресурс]: Uvicorn. URL: https://www.uvicorn.org/ (дата обращения: 02.05.2023).

4. Tortoise ORM

Tortoise ORM - это простой в использовании asyncio (асинхронный ввод-вывод) ORM (Object Relational Mapper) для Python. Он упрощает процесс взаимодействия с базами данных и предлагает поддержку различных баз данных, включая PostgreSQL[46]. Использование Tortoise ORM обеспечивает чистоту и поддерживаемость кодовой базы.

5. Aerich

Aerich - это инструмент миграции баз данных, специально разработанный для Tortoise ORM. Он позволяет беспрепятственно изменять схемы и управлять версиями, обеспечивая тем самым плавное развертывание и сопровождение приложения[47].

6. Pytest

Pytest - это многофункциональная среда тестирования для Python, которая позволяет легко писать и выполнять тесты. Обширный набор плагинов и мощные функции сделали его наиболее популярным инструментом тестирования python-приложений[48].

7. Loguru

Loguru - это библиотека, которая упрощает процесс логирования в приложениях Python. Она предлагает более понятный API, автоматическое форматирование журнала и расширенные возможности, такие как ротация журнала и обработка исключений[49].

8. Swagger UI

Swagger UI - это интерактивный инструмент документации для RESTful API, который позволяет пользователям визуализировать ресурсы API и

[46] Tortoise ORM [Электронный ресурс]: Tortoise ORM Documentation. URL: https://tortoise.github.io/ (дата обращения: 02.05.2023).

[47] Aerich [Электронный ресурс]: GitHub URL: https://github.com/tortoise/aerich (дата обращения: 02.05.2023).

[48] Pytest [Электронный ресурс]: Pytest documentation. URL: https://docs.pytest.org/en/latest/ (дата обращения: 02.05.2023).

[49] Loguru [Электронный ресурс]: GitHub URL: https://github.com/Delgan/loguru (дата обращения: 02.05.2023).

взаимодействовать с ними, не имея при этом фронтенда. Это способствует лучшему пониманию API и облегчает тестирование[50].

9 PostgreSQL

PostgreSQL - это мощная объектно-реляционная система баз данных с открытым исходным кодом, которая отличается надежностью, производительностью и расширенными возможностями, такими как поддержка JSON, полнотекстового поиска и пространственных данных[51]. Она широко используется для веб-приложений для хранения основных моделей данных платформы, таких как пользователи, оборудование и заказы.

10 MongoDB

MongoDB - это популярная база данных NoSQL с открытым исходным кодом, известная своей гибкостью, производительностью и масштабируемостью. Она хранит данные в JSON-подобном формате, что делает ее оптимальным выбором для работы с полуструктурированными данными, такими как сформированные документы[52].

11 Docker

Docker - это платформа, которая упрощает процесс создания, доставки и запуска приложений в контейнерах. Она обеспечивает согласованность и воспроизводимость в разных средах, что необходимо для беспрепятственного развертывания и масштабирования[53].

12 Docker Compose

Docker Compose - это инструмент для определения и запуска многоконтейнерных приложений Docker. Он позволяет разработчикам

[50] API Tools for Individuals, Teams, and Enterprise [Электронный ресурс]: Swagger. URL: https://swagger.io/tools (дата обращения: 02.05.2023).

[51] About PostgreSQL [Электронный ресурс]: PostgreSQL. URL: https://www.postgresql.org/about/ (дата обращения: 02.05.2023).

[52] What Is MongoDB? [Электронный ресурс]: MongoDB. URL: https://www.mongodb.com/what-is-mongodb (дата обращения: 02.05.2023).

[53] Developers bring their ideas to life with Docker [Электронный ресурс]: Docker. URL: https://www.docker.com/why-docker (дата обращения: 02.05.2023).

определить весь стек приложений, включая все сервисы и их зависимости, в одном файле, что упрощает процесс развертывания[54].

13 Flake8

Flake8 - это Python линтер, который проверяет стиль и качество кода, включая соблюдение PEP 8 (руководство по стилю Python), синтаксические ошибки и потенциальные ошибки[55]. Включение Flake8 в процесс разработки, поддерживает высокий уровень качества кода и помогает предотвратить появление ошибок или проблем, связанных с плохим стилем кодирования.

14 Black

Black - это популярный форматтер кода для Python, который обеспечивает единый стиль кода во всем проекте[56]. Использование Black, гарантирует, что весь код придерживается единого стиля, что облегчает его чтение и понимание. Такое единообразие упрощает сотрудничество между членами команды и помогает избежать недопонимания или конфликтов из-за различий в стилях кода.

15. Isort

Isort - это утилита Python, которая автоматически сортирует и упорядочивает импорты в файлах Python, обеспечивая правильный порядок импортов и снижая вероятность конфликтов слияния[57]. Используя isort, проект поддерживает чистоту и организованность кодовой базы, облегчая разработчикам навигацию и понимание структуры кода.

Данный стек технологий был выбран для программной реализации прототипа ЦПСИА по следующим причинам:

1. Производительность и масштабируемость. FastAPI, Uvicorn, PostgreSQL и MongoDB известны своей высокой

[54] Docker Compose overview [Электронный ресурс]: Docker Docs. URL: https://www.docker.com/why-docker (дата обращения: 02.05.2023).

[55] Flake8: Your Tool For Style Guide Enforcement [Электронный ресурс]: Flake8 Docs. URL: https://flake8.pycqa.org/en/latest/ (дата обращения: 02.05.2023).

[56] The uncompromising code formatter [Электронный ресурс]: Black Documentation. URL: https://black.readthedocs.io/en/stable/ (дата обращения: 02.05.2023).

[57] Home page [Электронный ресурс]: isort. URL: https://pycqa.github.io/isort/ (дата обращения: 02.05.2023).

производительностью и способностью эффективно обрабатывать большие объемы данных и запросов. Это гарантирует, что платформа может масштабироваться по мере необходимости без ущерба для производительности.

2. Ремонтопригодность и гибкость. Использование Python, Tortoise ORM, Aerich и Pytest гарантирует, что кодовая база является чистой, удобной для обслуживания и хорошо протестированной. Это облегчает добавление новых функций, исправление ошибок и адаптацию к изменяющимся требованиям.

3. Согласованность и воспроизводимость. Docker и Docker Compose обеспечивают последовательную сборку и запуск приложения в различных средах, что снижает вероятность возникновения проблем с развертыванием и упрощает управление жизненным циклом приложения.

4. Простота использования и совместная работа. Swagger UI предоставляет интерактивную документацию, которая облегчает понимание, тестирование и совместную работу над API. Это способствует эффективному общению между членами команды и упрощает процесс адаптации для новых разработчиков.

5. Стиль и качество кода. Поддержание единого стиля кода и обеспечение качества кода имеют решающее значение для долгосрочного успеха любого проекта. Использование flake8, black и isort в процессе разработки гарантирует, что кодовая база соответствует лучшим практикам и остается простой для понимания и сопровождения.

Таким образом, можно сказать, что выбранный стек технологий разработки должен обеспечить технические требования, указанные в пункте 1.2.

Структура реализованной серверной части прототипа ЦПСИА

Серверная часть прототипа ЦПСИА первого этапа текущего НИОКР можно разделить на несколько основных элементов: серверное приложение и 2 базы данных, – представленных в Приложении 6 (см. Приложение 6). Все указанные элементы запускаются в собственных Docker контейнерах.

Под серверным приложением подразумевается основная часть прототипа цифровой платформы, выполняющая весь функционал. В него входит Uvicorn ASGI-сервер и ядро серверного приложения прототипа ЦПСИА, которое было реализованно программно. Uvicorn сервер принимает запросы от клиентской части через HTTP/HTTPS протокол и передает их в ядро приложения. Ядро, в свою очередь, состоит из агрегатора (Router), реализованного на базе FastAPI, который связывает поступающие запросы от Uvicorn сервера с соответствующими модулями, выполняющими функционал бизнес-логики. Агрегатор представляет собой собранные собранные при запуске приложения контроллеры (*controllers*) каждого модуля (см. пункт 2.3.1). Модули могут обмениваться информацией с базами данных при обработке полученного запроса. Связь модулей между собой устанавливается на уровне агрегатора.

Помимо серверного приложения, в серверную часть прототипа ЦПСИА входят релятивистская (PostgreSQL) и нерелятивистская (MongoDB) базы данных. PostgreSQL база содержит данные, представляющие сущности основной бизнес-логики, а MongoDB используется для хранения файлов.

Реализация отдельных модулей прототипа ЦПСИА

Особенности программной реализации модуля авторизации и входа в личный кабинет

При программной реализации модуля авторизации и входа в личный кабинет была использована встроенная аутентификация FastAPI на основе OAuth2PasswordBearer и модуль jose для управления JWT токенами. Пароли хранятся в зашифрованном виде и обрабатываются с помощью CryptContext из passlib.context.

OAuth2PasswordBearer упрощает реализацию работы с паролями OAuth 2.0. OAuth 2.0 - это широко используемый и признанный стандарт для защиты API, обеспечивающий безопасный и контролируемый доступ к ресурсам. Поток паролей подходит для ситуаций, когда клиентское приложение имеет прямой доступ к учетным данным пользователя, как, например, в разрабатываемом прототипе ЦПСИА.

Библиотека jose является популярным выбором для работы с токенами JWT в Python. Она предоставляет набор удобных функций для создания, подписания, проверки и декодирования JWT-токенов, обеспечивая безопасный и надежный процесс аутентификации.

CryptContext использует безопасный алгоритм хэширования (выбран bcrypt) для хэширования паролей перед их сохранением в базе данных. Когда пользователи входят в систему, CryptContext может проверить введенный пароль по сохраненному хэшу, гарантируя, что введен правильный пароль, не раскрывая его открытым текстом. Использование CryptContext позволяет повысить безопасность разрабатываемого приложения и защитить конфиденциальные данные пользователей.

Особенности программной реализации модуля взаимодействия с данными из ФНС

Как было описано в пункте 2.5, модуль взаимодействия с данными из ФНС построен на базе сервиса DaData. Для программной реализации серверной части прототипа ЦПСИА использовалась Python библиотека[58], в частности, применялся API работы с ЕГРЮЛ и реестрами налоговой[59], позволяющий получать обширный перечень данных об организации по ИНН или ОГРН.

Особенности программной реализации модуля формирования стандартных форм договоров аренды

[58] Dadata API Client [Электронный ресурс]: GitHub. URL: https://github.com/hflabs/dadata-py (дата обращения: 02.05.2023).

[59] Организация по ИНН или ОГРН [Электронный ресурс]: DaData. URL: https://dadata.ru/api/find-party (дата обращения: 02.05.2023).

По своей структуре данный модуль формирования стандартных форм договоров аренды не отличается от типового модуля, добавление которого было описано в пункте 2.3 и не подразумевает использования специфических сторонних библиотек. Данный модуль содержит таблицу с текстами типовых договоров и актов, относящихся к допущенным на платформе сделкам. Основой функциональности модуля является извлечение подходящего шаблона из таблицы и заполнение его необходимыми данными, полученными из таблиц «Пользователь», «Компания» и «Оборудование». В конечном итоге пользователям предоставляется предзаполненный договор, версия для печати, а сам документ сохраняется в базе данных.

Особенности программной реализации модуля взаимной оценки и отзывов пользователей.

При программной реализации модуля взаимной оценки и отзывов пользователей была добавлена модель базы данных «Оценки» со следующими полями: от кого, кому, в рамках какой сделки, значение оценки, время и комментарий (опционально). Также потребовалось написать функции (сервис) для работы с этой таблицей (моделью), контроллер для их вызова и тесты. Следует отметить, что в процессе реализации данного модуля не возникла необходимость использования нестандартных сторонних библиотек.

Тестирование серверной части прототипа ЦПСИА

В процессе программной реализации согласно стандарту и спецификации, описанных в пункте 2.3, были написаны unit-тесты, позволяющие провести проверку корректности работы разрабатываемого модуля при различном полном спектре сценарий работы, включая штатную работы, вызовы исключений и граничные случаи. Минимальный уровень покрытия тестами был выбран 70%, и выполнялся во время программной реализации серверной части прототипа ЦПСИА.

Следует отметить, что в выбранный стек технологий разработки включён интерактивный инструмент документации Swagger UI (подробнее

описано в пункте 3.2.1), позволяющий проводить функциональное тестирование серверной части разрабатываемого прототипа цифровой платформы. Таким образом, кроме unit-тестов, было проведено ручное тестирование разработанного API для обмена данными с клиентской частью, что уменьшает риски некорректной работы разработанных модулей и облегчает тестирование взаимосвязи между разными модулям серверной части ЦПСИА. Интерфейса тестирования серверной части разрабатываемого прототипа цифровой платформы представлен в Приложении 7 (см. Приложение 7).

Целесообразно упомянуть, что тестирование и проверка выполнения технических параметрах, описанных в пункте 1.2, проводится после развертывания, когда определена аппаратная часть сервера и каналы передачи данных. Иными словами, проверка на выполнение данных параметров будет осуществлена при выполнения следующего этапа НИОКР.

Заключение

В рамках текущего этапа НИОКР были выполнены все цели и задачи в полном объеме:

1) Были сформированы основные функциональные, технические и архитектурные требования к прототипу ЦПСИА. Указанные требования относятся к финальной реализации прототипа ЦПСИА, выполнение которых позволит осуществить цели, стоящие перед разрабатываемым прототипом. Кроме того, были протестированы гипотезы о бизнес-модели и сформировано ценностное предложение, позволяющие определить путь пользователя, а следовательно принципы создания интерфейса и необходимых элементов разрабатываемого прототипа. Таким образом, были проанализированы основные аспекты, необходимые для дальнейшей разработки прототипа ЦПСИА.

2) Были разработаны основные элементы ЦПСИА, включая интерфейс, архитектуру клиентской и серверной частей, а также модулей авторизации, взаимной оценки, взаимодействия с данными из ФНС и модуля реализующего формирования стандартных форм договоров. Иными словами, была произведена проработка основных элементов прототипа разрабатываемой платформы, по которым можно сформировать техническое задание по программной реализации прототипа ЦПСИА.

3) Была осуществлена программная реализация интерфейса и серверной части прототипа ЦПСИА. В рамках этой задачи, на основе сформированных ранее требований, был выбран стек технологий разработки (набор языков программирования, библиотек и фреймворков), и написан код интерфейса и серверной части разрабатываемого прототипа цифровой платформы, включая настройку инфраструктуры, CI/CD и контейнеризацию.

Также были выполнены следующие работы, определённые в календарном плане: осуществлена формализация методологии оценки индексов лояльности (NPS) и удовлетворенности (CSI) пользователей, был реализован модуль по формированию стандартных формы договоров аренды;

были разработаны интерфейса и архитектура прототипа ЦПСИА, разработан стандарт и спецификация для внедрения новых модулей системы, модуль авторизации для входа в личный кабинет, модуль взаимодействия с данными из ФНС, модуль взаимной оценки и отзывов пользователей и серверная часть прототипа ЦПСИА; осуществлена программная реализация интерфейса прототипа ЦПСИА; программно реализована и протестирована серверная часть прототипа ЦПСИА.

Таким образом, все поставленные цели и задачи первого этапа НИОКР были выполнены в полном объеме.

Список источников

1. Бланк С., Дорф Б. Стартап: Настольная книга основателя / Стив Бланк, Боб Дорф; Пер. с англ. — М.: Альпина Паблишер, 2013. – 485 с. ISBN 978-5-9614-2809-4.
2. Гостилович А. О. Трансформация бизнес-моделей промышленных предприятий под воздействием экономики совместного потребления: дисс. … к-та экон. наук. М.: Интеллектуальная Система Тематического Исследования НАукометрических данных, 2022. 209 с
3. Дейкун В. С. СРАВНИТЕЛЬНЫЙ АНАЛИЗ МИКРОСЕРВИСНОЙ И МОНОЛИТНОЙ АРХИТЕКТУР //Студенческая наука: актуальные вопросы, достижения и инновации. – 2022. – С. 47-49.
4. Документация FSD [Электронный ресурс]: Feature-Sliced Design. URL: https://feature-sliced.design/ru/docs (дата обращения: 27.04.2023).
5. Закирова, Ю. М. Метод контейнеризации веб-приложений / Ю. М. Закирова // Приоритетные направления развития науки в современном мире : Сборник научных статей по материалам V Международной научно-практической конференции, Уфа, 05 февраля 2021 года. – Уфа: Общество с ограниченной ответственностью "Научно-издательский центр "Вестник науки", 2021. – С. 37-42. – EDN GXIXUO.
6. Кадыров, К. А. Микросервисная архитектура / К. А. Кадыров, А. М. Сафин // Фундаментальные и прикладные научные исследования: актуальные вопросы, достижения и инновации : сборник статей LII Международной научно-практической конференции : в 2 ч., Пенза, 15 января 2022 года. Том Часть 1. – Пенза: Наука и Просвещение (ИП Гуляев Г.Ю.), 2022. – С. 97-99. – EDN DWIUHS.
7. Кириллов В. В. Введение в реляционные базы данных. – БХВ-Петербург, 2012.
8. Логвинов, Д. В. Возможности отечественной облачной инфраструктуры на примере сервиса Yandex Cloud / Д. В. Логвинов, С. С. Савкин // Актуальные проблемы науки и образования в условиях современных вызовов : Сборник

материалов XIII Международной научно-практической конференции, Москва, 15 августа 2022 года. – Москва: Печатный цех, 2022. – С. 23-25. – EDN DMKKOK.

9. Маннанов, А. А. Разработка MVC паттерна для повышения эффективности разработки веб-приложений / А. А. Маннанов // Информационные технологии. Проблемы и решения. – 2019. – № 4(9). – С. 123-129. – EDN HEGPGX.
10. Мартишин С. А., Симонов В. Л., Храпченко М. В. Базы данных. Практическое применение СУБД SQL и NoSQL-типа для проектирования информационных систем. – 2017.
11. Михневич А. В. Микросервисная архитектура для разработки программного обеспечения. – 2022.
12. Мырзагалиев, А. М. Современные и адаптивные веб-приложения с поддержкой мобильных устройств / А. М. Мырзагалиев // Актуальные научные исследования в современном мире. – 2018. – № 1-1(33). – С. 25-30. – EDN YNLPYM.
13. Организация по ИНН или ОГРН [Электронный ресурс]: DaData. URL: https://dadata.ru/api/find-party (дата обращения: 02.05.2023).
14. Остервальдер А., Пинье И. Построение бизнес-моделей: Настольная книга стратега и новатора. Пер. с англ. – М. : Альпина Диджитал, - 330 с. ISBN:9785961423457
15. Рис Э. Бизнес с нуля: Метод Lean Startup для быстрого тестирования идей и выбора бизнес-модели / Эрик Рис ; Пер. с англ. – 8-е изд. – М. : Альпина Паблишер, 2022. – 255 с. ISBN 978-5-9614-6837-3
16. Ротарь В. Г., Ткачев М., Трофимова А. Е. Обзор актуальных фреймворков для разработки клиентской части современных веб-приложений //Молодежь и современные информационные технологии: сборник трудов XVIII Международной научно-практической конференции студентов, аспирантов и молодых учёных, 22-26 марта 2021 г., г. Томск. – Томский политехнический университет, 2021. – С. 170-171.

17. Сервис DaData [Электронный ресурс]: DaData. URL: https://dadata.ru/ (дата обращения: 02.05.2023).
18. Управляемые сервисы в облаке [Электронный ресурс]: Yandex Cloud. URL: https://cloud.yandex.ru/services (дата обращения: 03.05.2023).
19. Федотова, К. В. Обзор вариантов использования REST в современной архитектуре web-приложений / К. В. Федотова, А. Хоук // Безопасность городской среды : Материалы V Международной научно-практической конференции, Омск, 21–23 ноября 2017 года / Под ред. Е.Ю. Тюменцевой. – Омск: Омский государственный технический университет, 2018. – С. 410-412. – EDN USFJGD.
20. Федотова, К. В. Обзор вариантов использования REST в современной архитектуре web-приложений / К. В. Федотова, А. Хоук // Безопасность городской среды : Материалы V Международной научно-практической конференции, Омск, 21–23 ноября 2017 года / Под ред. Е.Ю. Тюменцевой. – Омск: Омский государственный технический университет, 2018. – С. 410-412. – EDN USFJGD.
21. Шарапов, С. Ф. Обзор способов разработки клиентских веб-приложений и преимущества использования генератора статичных сайтов / С. Ф. Шарапов // Информационные технологии и математическое моделирование (ИТММ-2021) : материалы XX Международной конференции имени А. Ф. Терпугова, Томск, 01–05 декабря 2021 года. – Томск: Национальный исследовательский Томский государственный университет, 2022. – С. 24-28. – EDN JUABOG.
22. About PostgreSQL [Электронный ресурс]: PostgreSQL. URL: https://www.postgresql.org/about/ (дата обращения: 02.05.2023).
23. Aerich [Электронный ресурс]: GitHub URL: https://github.com/tortoise/aerich (дата обращения: 02.05.2023).
24. Ahmed S., Mahmood Q. An authentication based scheme for applications using JSON web token //2019 22nd international multitopic conference (INMIC). – IEEE, 2019. – С. 1-6.

25. API Tools for Individuals, Teams, and Enterprise [Электронный ресурс]: Swagger. URL: https://swagger.io/tools (дата обращения: 02.05.2023).
26. Axios [Электронный ресурс]: GitHub. URL: https://github.com/axios/axios (дата обращения: 03.05.2023).
27. Da Silveira A. B., Hoppen N., De Camillis P. K. Flattening relations in the sharing economy: A framework to analyze users, digital platforms, and providers //Handbook of Research on the Platform Economy and the Evolution of E-Commerce. – IGI Global, 2022. – С. 26-51.
28. Dadata API Client [Электронный ресурс]: GitHub. URL: https://github.com/hflabs/dadata-py (дата обращения: 02.05.2023).
29. Damschroder L. J. et al. The updated Consolidated Framework for Implementation Research based on user feedback //Implementation Science. – 2022. – Т. 17. – №. 1. – С. 1-16.
30. Developers bring their ideas to life with Docker [Электронный ресурс]: Docker. URL: https://www.docker.com/why-docker (дата обращения: 02.05.2023).
31. Docker Compose overview [Электронный ресурс]: Docker Docs. URL: https://www.docker.com/why-docker (дата обращения: 02.05.2023).
32. Documentation [Электронный ресурс]: Sass. URL: https://sass-lang.com/documentation (дата обращения: 03.05.2023).
33. Eboli L., Mazulla G. (2009). A new customer satisfaction index for assessing the quality of transit services // Journal of Public Transport, 12 (3): pp. 21-37. – p. 3
34. FastAPI [Электронный ресурс]: fastapi.tiangolo URL: https://fastapi.tiangolo.com/ (дата обращения: 02.05.2023).
35. Feature Overview [Электронный ресурс]: React Router documentation. URL: https://reactrouter.com/en/6.11.1/start/overview (дата обращения: 03.05.2023).
36. Flake8: Your Tool For Style Guide Enforcement [Электронный ресурс]: Flake8 Docs. URL: https://flake8.pycqa.org/en/latest/ (дата обращения: 02.05.2023).
37. Getting Started [Электронный ресурс]: React documentation. URL: https://reactjs.org/docs/getting-started.html (дата обращения: 03.05.2023).

38. Getting Started [Электронный ресурс]: Vite documentation. URL: https://vitejs.dev/guide/ (дата обращения: 03.05.2023).
39. Getting Started with ESLint [Электронный ресурс]: ESLint documentation. URL: https://eslint.org/docs/user-guide/getting-started (дата обращения: 03.05.2023).
40. Getting Started with Redux [Электронный ресурс]: Redux documentation. URL: https://redux.js.org/introduction/getting-started (дата обращения: 03.05.2023).
41. Hill N., G. Brierley, and R. MacDougall. 2003. How to Measure Customer Satisfaction. Gower Publishing, Hampshire. pp. 12-16.
42. Home page [Электронный ресурс]: isort. URL: https://pycqa.github.io/isort/ (дата обращения: 02.05.2023).
43. JSON Web Token, [Электронный ресурс]: RFC 7519. URL: https://www.rfc-editor.org/rfc/rfc7519 (дата обращения: 27.04.2023).
44. Jsonwebtoken [Электронный ресурс]: GitHub. URL: https://github.com/auth0/node-jsonwebtoken (дата обращения: 03.05.2023).
45. Jukan A. et al. Lab 4—Connecting Edge and Cloud Tools with REST HTTP //Network of Things Engineering (NoTE) Lab. – Cham : Springer International Publishing, 2023. – С. 111-122.
46. Li L. et al. Design patterns and extensibility of REST API for networking applications //IEEE Transactions on Network and Service Management. – 2016. – Т. 13. – №. 1. – С. 154-167.
47. Loguru [Электронный ресурс]: GitHub URL: https://github.com/Delgan/loguru (дата обращения: 02.05.2023).
48. Overview [Электронный ресурс]: Formik docs. URL: https://formik.org/docs/overview (дата обращения: 03.05.2023).
49. Perrey R., Lycett M. Service-oriented architecture //2003 Symposium on Applications and the Internet Workshops, 2003. Proceedings. – IEEE, 2003. – С. 116-119.

50. Pytest [Электронный ресурс]: Pytest documentation. URL: https://docs.pytest.org/en/latest/ (дата обращения: 02.05.2023).
51. Reichheld F. «The one number you need to grow» // Harvard Business Review, vol. 81, no. 12, pp. 46-54, 2003.
52. The uncompromising code formatter [Электронный ресурс]: Black Documentation. URL: https://black.readthedocs.io/en/stable/ (дата обращения: 02.05.2023).
53. Tortoise ORM [Электронный ресурс]: Tortoise ORM Documentation. URL: https://tortoise.github.io/ (дата обращения: 02.05.2023).
54. Usage [Электронный ресурс]: Material-UI documentation. URL: https://mui.com/getting-started/usage (дата обращения: 03.05.2023).
55. Uvicorn [Электронный ресурс]: Uvicorn. URL: https://www.uvicorn.org/ (дата обращения: 02.05.2023).
56. Web Framework Benchmarks [Электронный ресурс]: Techempower. URL: https://www.techempower.com/benchmarks/ (дата обращения: 02.05.2023).
57. What Is MongoDB? [Электронный ресурс]: MongoDB. URL: https://www.mongodb.com/what-is-mongodb (дата обращения: 02.05.2023).
58. What’s New In Python 3.11, [Электронный ресурс]: Python Documentation. URL: https://docs.python.org/3/whatsnew/3.11.html (дата обращения: 02.05.2023).
59. 8 World-Class Software Companies That Use Python, [Электронный ресурс]: Real Python. URL: https://realpython.com/world-class-companies-using-python/ (дата обращения: 02.05.2023).

Реквизиты

Получатель гранта:
Общество с ограниченной ответственностью "ЦИФРОВАЯ ПЛАТФОРМА СОВМЕСТНОГО ИСПОЛЬЗОВАНИЯ АКТИВОВ" (ООО "ЦПСИА")

ИНН/КПП:7720876531/772001001

Юридический адрес:
111675, Г.Москва, ВН.ТЕР.Г. МУНИЦИПАЛЬНЫЙ ОКРУГ КОСИНО-УХТОМСКИЙ, УЛ ЛУХМАНОВСКАЯ, Д. 15, КВ. 47, 48

Фактический адрес:
111675, Г.Москва, ВН.ТЕР.Г. МУНИЦИПАЛЬНЫЙ ОКРУГ КОСИНО-УХТОМСКИЙ, УЛ ЛУХМАНОВСКАЯ, Д. 15, КВ. 47, 48

Тел: 89773776695 (моб.)

Банковские реквизиты:
Расчетный счет: 40702810538000181414 в банке ПАО СБЕРБАНК (Москва),
БИК 044525225, к/с 30101810400000000225

Генеральный директор

Гостилович А. О.

Москва, 2023