

Санкт-Петербургский государственный университет

Плоскин Александр Евгеньевич

Выпускная квалификационная работа

Разработка веб-приложения для работы с алгоритмами точного земледелия

Уровень образования: бакалавриат

Направление *02.03.03 “Математическое обеспечение и администрирование
информационных систем”*

Основная образовательная программа *СВ.5162.2020 “Технологии программирования”*

Научный руководитель:
старший преподаватель, к.т.н. Ю. В. Литвинов

Консультант:
инженер-исследователь, Mobile Robotics Lab, Skoltech А. В. Корнилова

Рецензент:
операционный директор ООО «Тензор Филдс» А. Ю. Петровская

Санкт-Петербург
2024

Saint Petersburg State University

Alexander Ploskin

Bachelor's Thesis

Web application for precision farming algorithms

Education level: bachelor

Speciality *02.03.03 “Software and Administration of Information Systems”*

Programme *CB.5162.2020 “Programming Technologies”*

Scientific supervisor:
C.Sc., senior lecturer Y.V. Litvinov

Consultant:
research engineer at Mobile Robotics Lab, Skoltech A.V. Kornilova

Reviewer:
Chief operating officer at Tensor Fields A.Y. Petrovskaya

Saint Petersburg
2024

Оглавление

Введение	4
1. Постановка задачи	6
2. Обзор	7
2.1. Обзор предметной области	7
2.2. Обзор аналогов	9
2.3. Обзор требований к реализуемому приложению	10
2.4. Обзор технологий для реализации приложения	11
3. Описание архитектуры	18
3.1. Архитектура серверной части	18
3.2. Архитектура клиентской части	20
4. Описание реализации	23
4.1. Пользовательские аккаунты	23
4.2. Карта	24
4.3. Редактирование границ полей	25
4.4. Визуализация показателей характеристик посевов	26
4.5. Заполнение данных севооборота и отображение урожайности	27
4.6. Локализация интерфейса	29
4.7. Тестирование	29
4.8. Непрерывное развертывание	30
5. Апробация	31
Заключение	33
Список литературы	34

Введение

В настоящее время часто обсуждаются вопросы изменения климата, увеличения мирового населения, ограниченности таких аграрных ресурсов, как пахотные земли и вода. Так, например, в отчете Продовольственной и сельскохозяйственной организации Объединенных наций [2] говорится о том, что 69% используемых водных ресурсов приходится на сельское хозяйство. В связи с этим становится очевидной актуальность проблемы повышения эффективности сельского хозяйства.

Распространенным подходом к увеличению эффективности сельского хозяйства является применение концепции точного земледелия. В работе [22] использование точного земледелия позволило увеличить урожайность пшеницы на 22%. Согласно статье [1] точное земледелие основано на использовании в управлении процессами земледелия таких показателей, как состояние почвы, здоровье посевов, действие удобрений и пестицидов, орошение, а также на применении технологий оценки и предсказания урожайности с высоким пространственным разрешением.

Одним из центров, разрабатывающих решения, связанные с точным земледелием, является Skoltech Agro (Сколковский институт науки и технологий, Россия)¹. В нем развиваются исследовательские проекты в сфере цифрового сельского хозяйства. Одним из таких проектов является Envirotyping and Digitalization². Его цель — разработать систему прогнозирования урожайности сельскохозяйственных культур с учетом факторов окружающей среды. Наличие качественного прогноза урожайности культур на полях помогает принимать такие решения, как выбор сельскохозяйственных культур и сортов, времени засева и сбора урожая, объема полива, финансовой политики предприятия, обладая большим количеством информации. Для реализации модели прогнозирования урожайности используются алгоритмы глубокого обучения, использующие временную компоненту (Spatio-Temporally Weighted Deep

¹<http://agro.skoltech.ru/> (дата обращения: 2024-05-02)

²<http://agro.skoltech.ru/projects1> (дата обращения: 2024-05-02)

Neural Networks), — подход к анализу временных рядов на многомерных данных. Это позволяет преодолеть недостатки существующих алгоритмов, которые для оценки урожайности используют только пространственную компоненту. В качестве входных данных для прогнозирования урожайности разрабатываемый инструмент использует информацию об урожайности в предыдущие годы, данные об окружающей среде, сценарии изменения климата, характерные для России свойства сельскохозяйственных культур. Реализацию решения можно разделить на две части — систему анализа данных, преобразующую данные для прогнозирования урожайности в предсказание, и приложение, цель которого состоит в том, чтобы предоставить пользователю интерфейс для ввода бизнес-данных и отображения прогнозов урожайности.

Данная работа посвящена разработке пользовательского приложения. В его функциональность входит: создание пользовательских аккаунтов, загрузка бизнес-данных (например, координаты полей, выращиваемые культуры, даты севооборота), отображение истории урожайности и предсказаний, визуализации другой необходимой информации о посевах. Для обеспечения универсального доступа к приложению с различных устройств без необходимости установки, а также облегчения обновлений, решение должно быть реализовано как веб-приложение.

1. Постановка задачи

Цель работы — реализовать веб-приложение для сбора и хранения пользовательских сельскохозяйственных бизнес-данных, вычисления показателей характеристик посевов и отображения предсказаний урожайности.

Для достижения цели были поставлены следующие задачи.

- Выполнить обзор существующих аналогичных решений.
- Сформулировать требования к приложению.
- Выполнить обзор технологий для создания веб-приложения, реализующих функциональность из требований.
- Разработать архитектуру приложения.
- Реализовать приложение согласно требованиям и архитектуре.
- Покрыть решение тестами.
- Реализовать непрерывное развертывание приложения.
- Провести апробацию решения.

2. Обзор

В этом разделе будет проведен обзор предметной области и аналогичных решений и выработаны требования к разрабатываемому приложению. Также на основе требований к приложению будет выполнен обзор основных технологий для его реализации.

2.1. Обзор предметной области

Цель разрабатываемого приложения — предоставить доступ к функциональности алгоритма предсказания урожайности и к необходимым показателям характеристик посевов. Для использования алгоритма предсказания урожайности необходимо получить от пользователя следующие данные:

- пространственные координаты полей. Может быть использован ручной ввод границ на карте, извлечение данных из файлов, а также автоматическое определение границ по спутниковым изображениям. Согласно данным Росстата [64] средняя площадь полей в сельскохозяйственной организации составила 351,6 гектара, из этого следует необходимость автоматизации ввода границ полей;
- изображения полей в определенный момент времени. Согласно [38] может быть использована либо аэрофотосъемка, либо спутниковые изображения. Например, источник спутниковых снимков Sentinel-2 MSI³ предоставляет изображения раз в десять дней и имеет разрешение от 10 до 60 метров. Использование аэрофотосъемки позволяет работать с изображениями в более высоком разрешении и в необходимые для пользователя даты, но имеет большую стоимость;
- даты севооборота и выращиваемые культуры и сорта. Работа алгоритма предсказания урожайности связана с выращиваемой культурой, помимо этого на результаты влияет используемый сорт,

³<https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-2-msi/overview> (дата обращения: 2024-05-02)

каждый из которых имеет уникальные свойства. Использование дат севооборота позволяет отслеживать показатели характеристик посевов в различные стадии роста;

- историю урожайности культуры в предыдущие сезоны.

Для оценки состояния посевов и принятия хозяйственных решений рассматриваются различные показатели. Чаще всего рассматриваются следующие индексы:

- NDVI (Normalized Difference Vegetation Index), согласно [66] NDVI — это числовой показатель качества и количества растительности на участке поля, используемый для отслеживания состояния посевов в различные этапы сезона. На рис. 1 можно ознакомиться с примером соотношения величины NDVI с внешним видом растений;
- NDMI (Normalized Difference Moisture Index), согласно [28] NDMI — это числовой показатель содержания влаги в растениях на участке поля;

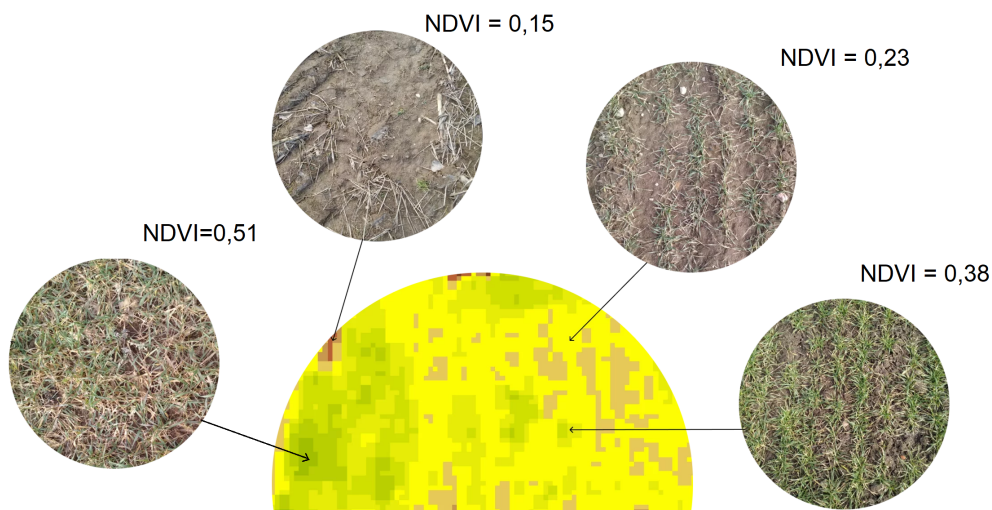


Рис. 1: Пример соотношения величины NDVI с состоянием растений на поле из статьи [66]

2.2. Обзор аналогов

Разрабатываемое приложение предоставляет инновационный алгоритм предсказания урожайности, что делает его уникальным. Далее будут рассмотрены наиболее популярные приложения на рынке точного земледелия, также работающие с земледельческой информацией и отображающие урожайность и показатели характеристик посевов. В обзоре уделено внимание реализации работы с пользовательскими данными и вычисления индексов.

2.2.1. OneSoil

OneSoil [31] — бесплатное приложение для использования методов точного земледелия. Предоставляет пользователю как веб-интерфейс, так и мобильное приложение. В числе прочих предоставляет пользователю следующие функции:

- задание границ полей вручную, через загрузку файлов в KML формате [20] и выбором на карте автоматически определенных границ;
- выявление облаков на спутниковых снимках;
- отображение таких индексов, как NDVI, средний NDVI и неоднородность NDVI;
- возможность сравнения значений индексов на поле в две разные даты;
- отображение значений индексов как на карте, так и на графике;
- уведомление пользователей о появлении новых снимков полей.

2.2.2. Exact Farming

Exact Farming [65] — коммерческая платформа для управления агробизнесом, предоставляющая следующие функции точного земледелия:

- задание границ полей вручную и через загрузку KML-файлов;
- возможность использовать как спутниковые снимки, так и БПЛА-аэрофотосъемку для вычисления показателей характеристик посевов;
- отображение следующих характеристик посевов: NDMI, NDVI и его модификаций EVI⁴ и MSAVI⁵;
- отображение значений индексов на карте и на графике.

2.3. Обзор требований к реализуемому приложению

Исходя из потребностей модели предсказания урожайности, успешных решений в аналогах и видения предметной области были выдвинуты следующие функциональные требования к приложению.

- Возможность создания пользовательских аккаунтов и авторизации.
- Отображение интерактивной карты с возможностью поиска по географическим названиям и координатам.
- Создание полей с группировкой их в дивизионы, задание границ вручную и через загрузку файлов в KML формате.
- Создание сезонов и заполнение данных севооборота для конкретного сезона и поля, включая площадь посева, культуру и сорт, урожайность, даты посева и сбора урожая.
- Визуализация истории урожайности и ее прогноза на графике с использованием модели предсказания урожайности.
- Отображение следующих показателей характеристик посевов и их средних значений: NDVI, NDMI.

⁴<https://satyukt.medium.com/enhanced-vegetation-index-evi-and-its-significance-in-agriculture-monitoring-6d9a6c3e71c0> (дата обращения: 2024-05-02)

⁵<https://pro.arcgis.com/ru/pro-app/latest/arcpy/image-analyst/msavi.htm> (дата обращения: 2024-05-02)

- Приложение должно предоставлять возможность выбрать русский или английский в качестве языка интерфейса.

Помимо функциональных требований, были также выдвинуты следующие системные ограничения и нефункциональные требования:

- архитектурное разделение клиентской и серверной частей для возможности создания мобильного приложения,
- покрытие кода модульными тестами,
- автоматическое развертывание и тестирование приложения,
- соблюдение единого стандарта оформления кода в клиентской и серверной частях.

2.4. Обзор технологий для реализации приложения

Для реализации приложения согласно сформулированным требованиям необходимо выбрать технологии для создания клиентской и серверной частей веб-приложения, а также инструменты для поддержки стандартов оформления кода.

2.4.1. Обзор технологий для реализации серверной части

Для ускорения разработки, обеспечения безопасности и эффективности реализации серверной части приложения было принято решение основывать реализацию на существующем фреймворке. Исходя из потребностей приложения к нему были выдвинуты следующие требования:

- простота эффективной интеграции существующего кода алгоритма предсказания урожайности, написанного на Python;
- возможность писать код на одном из известных разработчику языков, включая Python, C#, JavaScript, Java;
- популярность в сообществе;

- простота интеграции с базой данных за счет наличия встроенной библиотеки, реализующей объектно-реляционное отображение (англ. Object-Relational Mapping, ORM), и кеширования запросов;
- встроенные функции для работы с пользовательскими аккаунтами;
- высокая производительность.

В обзор включены фреймворки, для которых число подписчиков на соответствующие им темы на GitHub⁶ более десяти тысяч в июле 2023 года, а именно: Node.js (74 тысячи), Django (54 тысячи), ASP.NET (44 тысячи), Spring boot (28 тысяч), Flask (27 тысяч). Laravel и Ruby on rails не были включены в обзор, так как требуют написания кода на языках PHP и Ruby соответственно.

На основе сформулированных требований был проведен сравнительный анализ выбранных вариантов.

- Django [7] является фреймворком для разработки приложений на языке Python, что обеспечивает нативную интеграцию с кодом алгоритма предсказания урожайности, имеет встроенную ORM, функции кеширования и создания пользовательских аккаунтов. Недостатками является ограниченность языка Python для создания приложений со сложной бизнес-логикой [58].
- Flask [10] также предоставляет Python как язык разработки, но имеет ограниченную функциональность и не имеет встроенных ORM, создания пользовательских аккаунтов, валидации и кеширования.
- ASP.NET [3] — фреймворк для создания веб-приложений на платформе .NET, предоставляет встроенные функции для создания аккаунтов и управления сессиями, имеет встроенную ORM и функ-

⁶<https://github.com/> (дата обращения: 2024-05-02)

ции кеширования. Согласно [51] имеет лучшую производительность, чем Django.

- Node.js [29] позволяет писать код приложения как на JavaScript, так и на других языках, которые могут быть скомпилированы в JavaScript, что предоставляет возможность иметь единый язык для написания кода клиентского и серверного приложений. Необходимая функциональность доступна через сторонние библиотеки. Согласно [51] имеет лучшую производительность, чем Django.
- Spring boot [48] — фреймворк для создания веб-приложений для платформы JVM, имеет встроенную ORM и поддержку других необходимых функций. Согласно статье [45] уступает в производительности Node.js.

Для платформ без нативной интеграции кода на Python, алгоритм предсказания урожайности может быть интегрирован через сторонние библиотеки, межпроцессорное взаимодействие или же через связь с отдельным сервисом, написанным на Django или Flask. Простота интеграции и единообразие кодовой базы серверной части приложения является большим преимуществом Django по сравнению с другими платформами. Большая часть вычислительной сложности относится к алгоритмам вычисления характеристик посевов и предсказания урожайности, которые уже реализованы на языке Python, следовательно, прирост производительности из-за выбора другого фреймворка не был бы значительным. Таким образом, Django был выбран как фреймворк для написания серверной части приложения.

Для хранения и управления данными приложения необходима СУБД (система управления базами данных). Выбор пал в пользу локальной СУБД с реляционной моделью данных, так как структура данных в приложении четко определена, а размещение базы на одном компьютере проще при разработке. Среди СУБД с такими свойствами была выбрана PostgreSQL [36], она бесплатна для использования, имеет открытый исходный код и поддерживает требования ACID⁷. В

⁷<https://en.wikipedia.org/wiki/ACID> (дата обращения: 2024-05-02)

то же время, в отличие от SQLite [46], она предоставляется как SaaS (англ, Software as a Service) [57] большинством облачных платформ, например, Yandex Cloud [37]. Преимуществом PostgreSQL над MySQL [27] в рамках разрабатываемого приложения является лучшая поддержка хранения и индексирования геопространственных и растровых данных, предоставляемая инструментом PostGIS [35]. Для связи PostgreSQL с Django ORM был выбран плагин Psycopg⁸, он изолирует в себе функциональность подключения к базе данных и обеспечивает потокобезопасность.

Для поддержания оформления кода в соответствии со стандартом PEP 8⁹ и выявления ошибок в коде статическим анализом были выбраны инструменты Black¹⁰ и Flake8¹¹. Для оформления импортов пакетов используется инструмент isort¹². Управление внешними зависимостями осуществляется при помощи pip-tools¹³. Автоматический запуск работы инструментов перед каждым коммитом выполняется при помощи фреймворка pre-commit¹⁴.

2.4.2. Обзор технологий для реализации клиентской части

Веб-фреймворк Django предоставляет инструмент для реализации клиентской части приложения Django template engine¹⁵. Его использование позволило бы использовать встроенные функции и упростило бы интеграцию с кодом серверной части. Однако это нарушило бы требование о разделении клиентской и серверной частей. Исходя из потребностей приложения к фреймворку для написания клиентской части были выдвинуты следующие требования:

- популярность в сообществе;

⁸<https://www.psycopg.org/docs/> (дата обращения: 2024-05-02)

⁹<https://peps.python.org/pep-0008/> (дата обращения: 2024-05-02)

¹⁰<https://github.com/psf/black> (дата обращения: 2024-05-02)

¹¹<https://flake8.pycqa.org/en/latest/> (дата обращения: 2024-05-02)

¹²<https://pycqa.github.io/isort/> (дата обращения: 2024-05-02)

¹³<https://github.com/jazzband/pip-tools> (дата обращения: 2024-05-02)

¹⁴<https://pre-commit.com/> (дата обращения: 2024-05-02)

¹⁵<https://docs.djangoproject.com/en/4.2/topics/templates/> (дата обращения: 2024-05-02)

- использование JavaScript;
- возможность повторного использования кода для написания мобильного приложения;
- функции маршрутизации и управления состоянием;
- высокая производительность;
- небольшой размер библиотеки для быстрой загрузки страницы пользователем.

На основе требований был проведен сравнительный анализ фреймворков, имеющих более сорока тысяч подписчиков на соответствующие им темы на GitHub в июле 2023 года.

- React [39] является наиболее популярным фреймворком в сообществе согласно сравнению [26], платформа React Native [40] позволяет переиспользовать код для создания мобильного приложения, имеет сравнительно небольшой размер в 40 КБ. К недостаткам можно отнести необходимость использования сторонних библиотек для реализации маршрутизации и управления состоянием.
- Angular [4] предоставляет все необходимые функции для создания приложения, повторное использование кода для написания мобильной версии доступно только через сторонние платформы, например, Ionic¹⁶. Однако имеет сравнительно большой размер в 143 КБ, и относительно невысокую производительность [17].
- Vue.js [56] имеет наименьший размер из рассмотренных вариантов в 21 КБ и самую высокую производительность [17] среди рассматриваемых фреймворков. Так как для Vue Native¹⁷ не публикуются обновления, переиспользование кода для разработки мобильного приложения возможно только через сторонние платформы.

¹⁶<https://ionicframework.com/docs> (дата обращения: 2024-05-02)

¹⁷<https://vue-native.io/> (дата обращения: 2024-05-02)

Исходя из большей популярности React и возможности повторного использования кода в мобильном приложении, этот фреймворк был выбран для разработки клиентской части приложения. В качестве языка разработки был выбран TypeScript [53], так как он расширяет JavaScript статической типизацией. Для поддержания единого стандарта оформления кода клиентской части, следуя дорожной карте Microsoft [52], был выбран ESLint¹⁸.

2.4.3. Обзор платформ для работы с картографией

Важным компонентом клиентской части приложения является глобальная карта. На ней пользователь создает, редактирует и просматривает границы полей, и изучает показатели характеристик посевов. Исходя из потребностей приложения были выдвинуты следующие требования к библиотеке для работы с глобальной картой:

- возможность использования различных источников спутниковых карт, в том числе Google Maps¹⁹, ESRI Satellite²⁰, Maxar²¹ и Bing Maps²²;
- встроенная реализации навигации по карте и изменения масштаба;
- возможность добавлять текст, графические маркеры, полигоны и изображения поверх карты;
- наличие лицензии, допускающей бесплатное использование;
- активная поддержка, последнее обновление должно быть не позднее мая 2023 года.

¹⁸<https://eslint.org/> (дата обращения: 2024-05-02)

¹⁹<https://mapsplatform.google.com/> (дата обращения: 2024-05-02)

²⁰<https://www.arcgis.com/home/item.html?id=10df2279f9684e4a9f6a7f08febac2a9> (дата обращения: 2024-05-02)

²¹<https://www.maxar.com/> (дата обращения: 2024-05-02)

²²<https://www.microsoft.com/en-us/maps/bing-maps/> (дата обращения: 2024-05-02)

Платформы Google Maps API [13] и Bing Maps API [5] позволяют работать только с одним источником спутниковых карт. Mapbox [23] является платным решением. Помимо этих платформ исходя из сформулированных требований были также рассмотрены решения, представленные ниже.

- Leaflet [21] является библиотекой с открытым исходным кодом и лицензией, допускающей коммерческое использование. Богатая коллекция плагинов обеспечивает всю необходимую функциональность, в качестве источника данных может быть использована любая коллекция, предоставляющая спутниковые снимки.
- OpenLayers [32] также имеет открытый исходный код, допускает коммерческое использование и предоставляет необходимую функциональность. Имеет меньшую, по сравнению с Leaflet, популярность (40 тысяч звезд на репозитории против 10 тысяч) и согласно [33] более низкую производительность.
- Maplibre [60] — библиотека с открытым исходным кодом, представляющая собой ответвление от Mapbox после ее перехода на проприетарную лицензию. Предоставляет всю необходимую функциональность, однако значительно уступает Leaflet в популярности (40 тысяч звезд на репозитории против 5 тысяч).

Таким образом, Leaflet была выбрана библиотекой для работы с глобальной картой.

3. Описание архитектуры

Для реализации приложения на стороне сервера был выбран фреймворк Django, на стороне клиента — React. В этом разделе будет описана глобальная архитектура приложения, включающая аспекты взаимодействия между клиентом и сервером, архитектуру клиентской и серверной частей приложения.

3.1. Архитектура серверной части

При проектировании серверной части веб-приложения одним из важных решений является выбор между микросервисной и монолитной архитектурой. Согласно статье [25] приложения с монолитной архитектурой реализуют всю бизнес-логику в одной кодовой базе, не зависящей от других приложений. В то же время, приложения с микросервисной архитектурой реализуют бизнес-логику на независимо развертываемых сервисах, каждый из которых реализует определенную ее часть, и взаимодействуют друг с другом по сети. Разница между двумя упомянутыми архитектурными подходами проиллюстрирована на рис. 2. Функциональность приложений с монолитной архитектурой сложнее масштабировать, а сами приложения менее устойчивы при увеличении нагрузки или возникновении ошибок. Тем не менее они проще при разработке, так как не требуют написания логики взаимодействия между различными сервисами, и легче при развертывании. Исходя из этих соображений, приложение реализовано в соответствии с монолитной архитектурой.

Для того чтобы клиент мог использовать функции серверной части приложения, ему предоставляется веб-интерфейс, основанный на архитектурном стиле REST. Согласно принципам и ограничениям REST [9] взаимодействие клиента с сервером осуществляется при помощи стандартных методов HTTP (GET, POST, PUT, DELETE), при этом сервер не хранит состояние сессии, что обеспечивает однозначную идентифицируемость запроса и прозрачную архитектуру взаимодействия. Для реализации REST API в приложении используется платформа Django

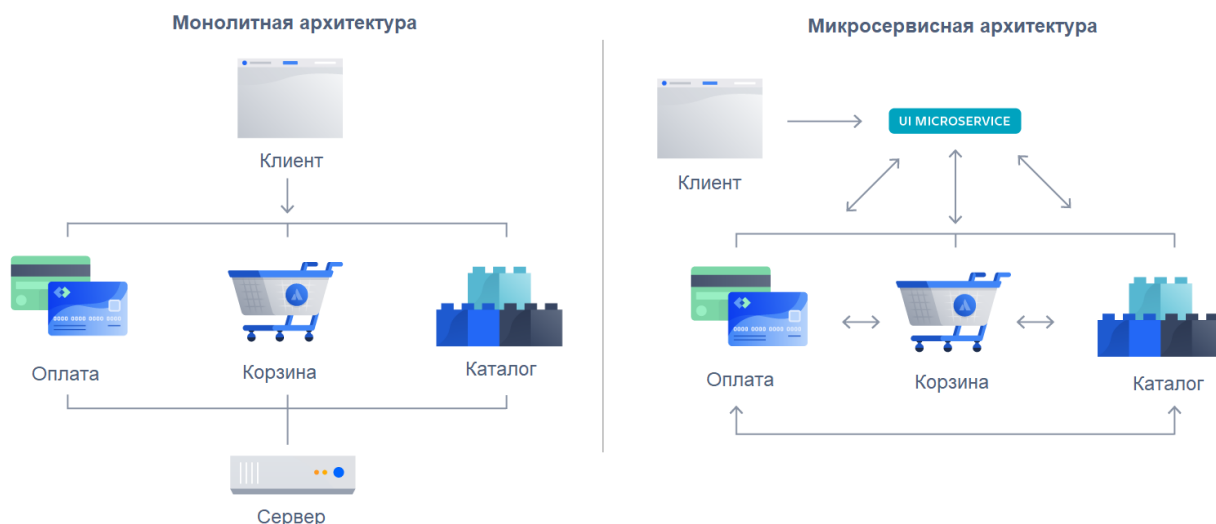


Рис. 2: Схемы монолитной и микросервисной архитектур из [25]

REST framework²³, предоставляющая функции авторизации, ограничения доступа к функциям интерфейса, сериализации и десериализации, валидации, кеширования, пагинации, а также создания ответов на запросы к интерфейсу. Для удовлетворения запроса клиента функция компонента REST API обращается к компоненту бизнес-логики, который реализует функции управления данными в соответствии с алгоритмами предметной области, взаимодействует с моделью предсказания урожайности и манипулирует базой данных через уровень доступа к данным, реализуемый Django ORM. Логика обработки клиентского запроса проиллюстрирована на рис. 3.

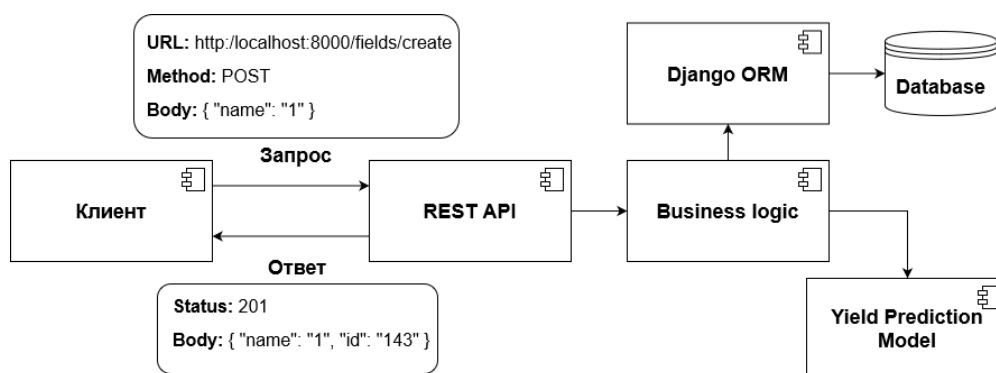


Рис. 3: Диаграмма компонентов серверной части приложения

Обработка клиентских запросов к серверной части приложения

²³<https://www.django-rest-framework.org/> (дата обращения: 2024-05-02)

осуществляется при помощи веб-сервера nginx [61], он получает запросы, обрабатывает их в неблокирующем режиме и отправляет ответ. Для взаимодействия веб-сервера с кодом приложения используется интерфейс шлюза веб-сервера (англ. Web Server Gateway Interface, WSGI) [34]. WSGI-сервер реализует WSGI и отделяет код сервера от кода приложения. В качестве WSGI-сервера для приложения был выбран Gunicorn [14]. Он работает по pre-fork модели — главный поток обрабатывает все входящие запросы и распределяет их между рабочими потоками. Это позволяет обойти ограничение интерпретатора языка Python [12] и обрабатывать запросы параллельно и без блокировок сервера. Процесс обработки запросов на уровне веб-сервера проиллюстрирован на рис. 4.

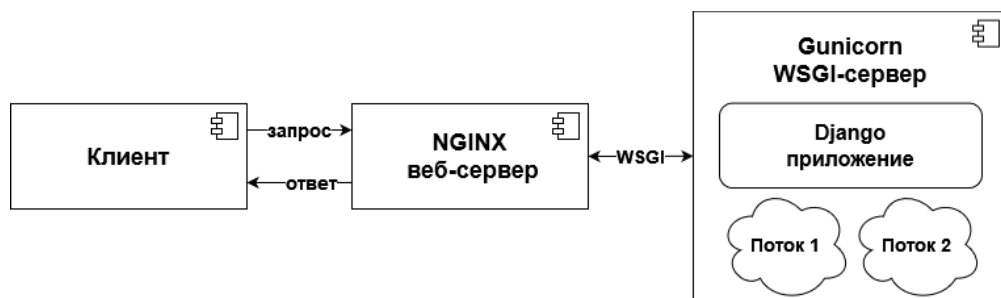


Рис. 4: Схема обработки клиентских запросов на уровне веб-сервера

3.2. Архитектура клиентской части

Клиентская часть приложения спроектирована как одностраничное приложение (англ. Single Page Application, SPA) [47]. Данный подход заключается в том, что весь код клиентского приложения компилируется в одну веб-страницу, представленную в виде набора статических файлов, которые доставляются пользователю при загрузке начальной страницы. Процесс компиляции исходного кода React-приложения проиллюстрирован на рис. 5 и включает в себя такие этапы как:

- установка сторонних пакетов утилитой npm²⁴;

²⁴<https://www.npmjs.com/> (дата обращения: 2024-05-02)

- компиляция исходного кода приложения при помощи Babel²⁵ и Typescript²⁶;
- преобразование модулей в набор статических файлов сборщиком webpack²⁷.

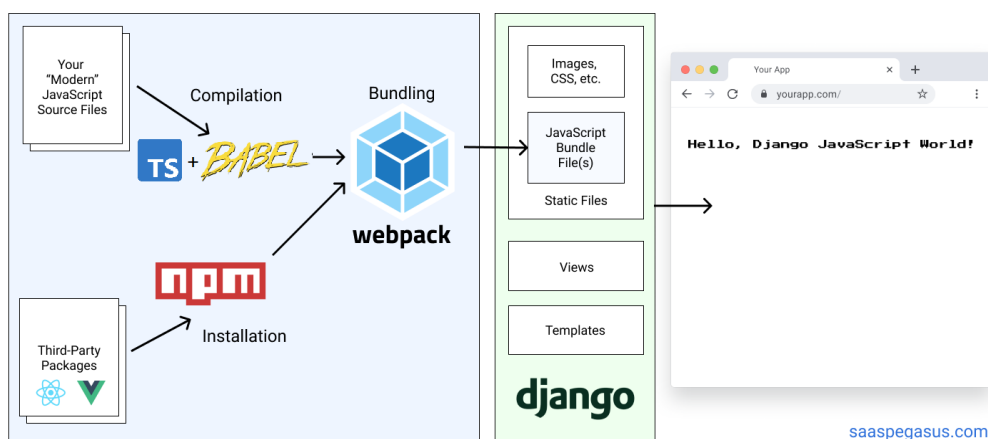


Рис. 5: Схема процесса компиляции клиентского приложения из [15]

После начала работы в браузере приложение обращается к серверу через REST API только для получения и модификации данных. Маршрутизация осуществляется на уровне клиентского приложения и реализована при помощи библиотеки React Router²⁸. Такая архитектура обеспечивает разделение логики серверной и клиентской частей приложения и уменьшает нагрузку на сервер за счет уменьшения числа запросов.

Управление состоянием приложения реализовано через использование библиотеки Redux [42]. Интерфейс приложения обрабатывает действия пользователя и преобразует их в объекты типа **Action**, содержащие в себе информацию о событии. После этого соответствующие событию функции типа **Reducer** преобразуют относящуюся к ним часть состояния приложения, хранящуюся в объекте типа **Storage**. Далее объект виртуального дерева React рисуется заново в соответствии с измененным состоянием приложения. Взаимосвязь между хранящимся

²⁵<https://babeljs.io/> (дата обращения: 2024-05-02)

²⁶<https://www.typescriptlang.org/> (дата обращения: 2024-05-02)

²⁷<https://webpack.js.org/> (дата обращения: 2024-05-02)

²⁸<https://reactrouter.com/en/main> (дата обращения: 2024-05-02)

в памяти состоянием приложения и компонентами виртуального дерева React реализуется библиотекой React Redux [41]. Описанная схема работы проиллюстрирована на рис. 6.

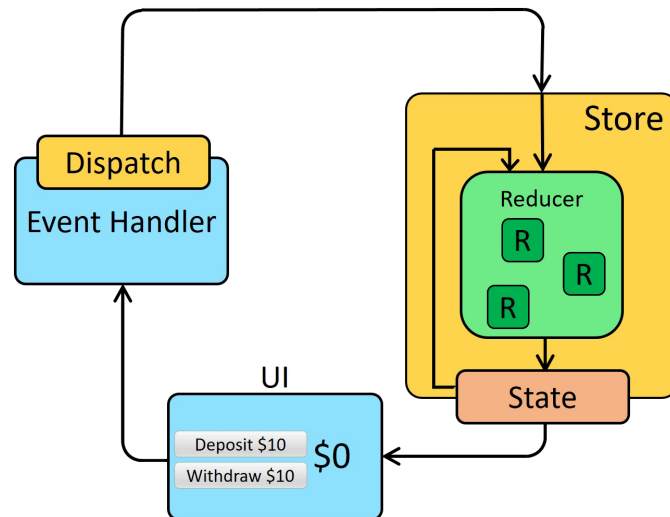


Рис. 6: Схема изменения данных в приложении [43]

Логика выполнения запросов к серверу, кеширования результатов запросов, а также политика их выполнения, включая авторизацию, обработку ошибок, повторение запросов и защиту от CSRF-атак [6], осуществляется при помощи функций библиотеки Redux Toolkit [44].

4. Описание реализации

Разработанная архитектура осуществляет разделение кода приложения на уровни пользовательского интерфейса, бизнес-логики и работы с данными. В данном разделе приведены детали реализации функциональности системы на различных уровнях в соответствии с предложенными требованиями.

4.1. Пользовательские аккаунты

Для обеспечения долговременного хранения информации о хозяйствах необходима функциональность пользовательских аккаунтов. Django предоставляет инструменты для создания и хранения информации о пользователях. Для авторизации запросов при использовании REST API в приложении применяется стандарт JSON Web Token (JWT) [18]. JWT представляет собой токен, содержащий зашифрованную информацию о пользователе и его правах. При предъявлении логина и пароля сервер предоставляет токен доступа (access-токен), который включается в заголовок каждого запроса. Для улучшения безопасности системы access-токен имеет ограниченное время жизни, после его истечения клиент запрашивает новый access-токен, используя полученный ранее токен обновления (refresh-токен). Описанный алгоритм авторизации проиллюстрирован на рис. 7. Для реализации JWT-авторизации на стороне сервера используется пакет `djoser` [59]. Он предлагает функции создания и управления пользователями, включая регистрацию, аутентификацию и восстановление пароля.

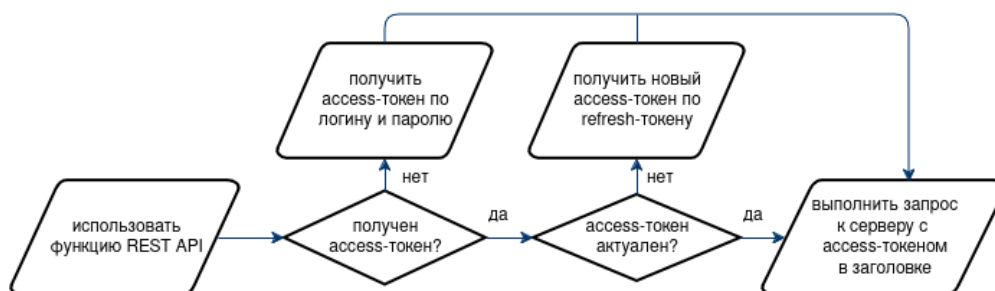


Рис. 7: Блок-схема алгоритма JWT-авторизации

4.2. Карта

Компонент карты служит для создания, редактирования и отображения границ полей в хозяйстве пользователя, и визуализации показателей характеристик посевов. Подложкой карты служит совокупность спутниковых снимков Google Earth [13], отображаемых на плоскость проекцией EPSG:3857²⁹, с наложенной поверх них географической информацией. Используемая библиотека Leaflet [21] загружает спутниковую карту через API Google Maps в виде отдельных фрагментов размером 256 на 256 пикселей в зависимости от текущего масштаба, схема разделения карты на фрагменты проиллюстрирована на рис. 8.

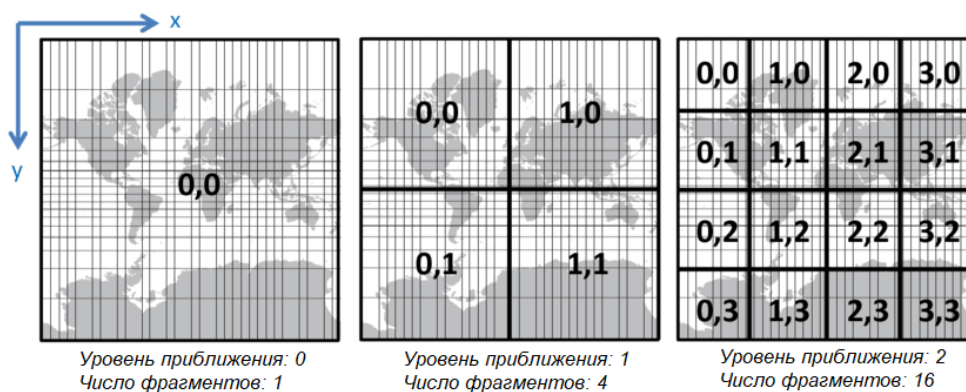


Рис. 8: Схема разделения карты на фрагменты в Google Maps [49]

Для упрощения навигации пользователя на карте реализован поиск по названиям географических объектов и координатам. В качестве поискового движка используется Nominatim [30], он предоставляет API для поиска географических объектов среди данных открытого источника OpenStreetMap³⁰, а также может быть развернут на собственном сервере. Связь поискового движка с картой Leaflet реализована при помощи плагина leaflet-geosearch³¹, помимо этого, он предоставляет функцию автозаполнения поля ввода названия географического объекта.

²⁹https://wiki.openstreetmap.org/wiki/Web_Mercator (дата обращения: 2024-05-02)

³⁰<https://www.openstreetmap.org/about> (дата обращения: 2024-05-02)

³¹<https://github.com/smeijer/leaflet-geosearch> (дата обращения: 2024-05-02)

4.3. Редактирование границ полей

Для реализации редактирования границ полей на карте в приложении используется плагин `leaflet-geoman`³². Этот плагин предоставляет возможности создания и редактирования полигонов, а также проверку на корректность полученных границ, например, запрет пересечения граней полигона. Пример редактирования границ поля в приложении показан на рис. 9.



Рис. 9: Редактирование границ поля

Помимо ручного редактирования границ полей, для ввода большого числа данных могут быть использованы файлы в формате KML [20]. Формат KML основан на XML³³ и содержит в себе сгруппированную по географическим объектам информацию об их координатах. Разбор файлов осуществляется при помощи методов стандартной библиотеки Python³⁴. Пример загруженных через файл в приложение данных фермерского хозяйства с восьмьюдесятью полями показан на рис. 10.

³²<https://www.geoman.io/> (дата обращения: 2024-05-02)

³³<https://en.wikipedia.org/wiki/XML> (дата обращения: 2024-05-02)

³⁴<https://docs.python.org/3/library/xml.etree.elementtree.html> (дата обращения: 2024-05-02)

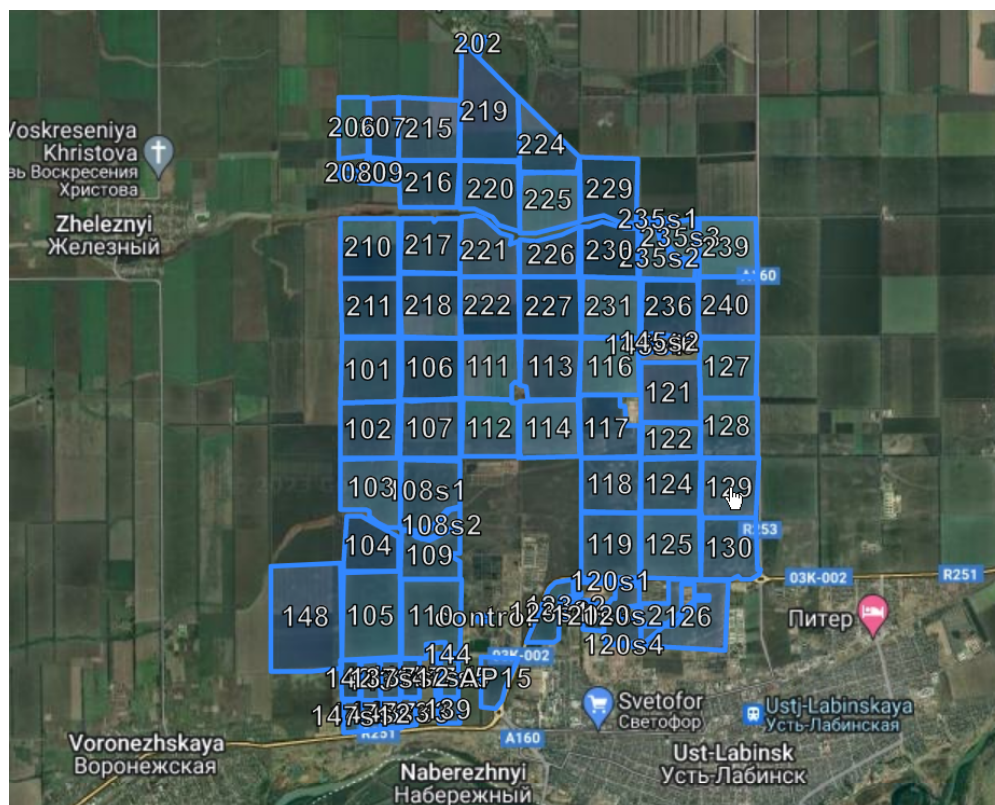


Рис. 10: Загруженные в приложение данные фермерского хозяйства

4.4. Визуализация показателей характеристик посевов

Приложение позволяет пользователям визуализировать такие показатели характеристик посевов, как NDVI, NDMI. Значения этих индексов вычисляются в серверной части приложения по соответствующим формулам на основе изображений, предоставляемых провайдером Sentinel-2 MSI [54]. Это коллекция спутниковых изображений, обновляющаяся каждые десять дней, каждое изображение включает в себя множество компонентов, характеризующихся диапазоном длин отраженной волны, и имеющих различное пространственное разрешение. В число компонентов, в том числе, входят инфракрасный диапазон (NIR) с разрешением в десять метров, красный диапазон (RED) также с разрешением в десять метров и коротковолновой инфракрасный диапазон (SWIR) с разрешением в двадцать метров. Для вычисления NDVI по этим компонентам применяется формула:

$$NDVI = \frac{NIR - RED}{NIR + RED}$$

Показатель NDMI вычисляется по формуле:

$$NDMI = \frac{NIR - SWIR}{NIR + SWIR}$$

Для фильтрации изображений по координатам и датам съемки, соединения нескольких снимков, определения границ облаков и вычисления значений индексов по формулам используется интерфейс платформы Google Earth Engine [55].

Для визуализации значений индексов на карте клиент загружает с сервера маску для поля в формате PNG³⁵, пиксели которой соответствуют цветовой интерпретации значения. Полученное изображение накладывается поверх карты при помощи функций библиотеки Leaflet. Для сопоставления цвета пикселя значению индекса клиент дополнительно получает маску в формате GeoTIFF³⁶ и выводит на экран значение индекса в выбранной координате. Парсинг GeoTIFF-маски в клиентском приложении осуществляется при помощи библиотеки georaster³⁷. С примером визуализации NDVI для поля в приложении можно ознакомиться на рис. 11.

4.5. Заполнение данных севооборота и отображение урожайности

Для предоставления алгоритму предсказания урожайности необходимых данных клиентское приложение предлагает пользователю интерфейс для заполнения данных севооборота. Для выбранного сезона и поля пользователь может ввести информацию о площади и датах посева и сбора урожая, урожайности с гектара и выращиваемой культуре и ее сорте. Список возможных культур задан в приложении и редактируется пользователями с правами администратора благодаря механизму Django admin site³⁸. На линейный график выводится история урожайности культуры на поле в предыдущие сезоны и прогноз на будущее.

³⁵<https://en.wikipedia.org/wiki/PNG> (дата обращения: 2024-05-02)

³⁶<https://en.wikipedia.org/wiki/GeoTIFF> (дата обращения: 2024-05-02)

³⁷<https://github.com/GeoTIFF/georaster> (дата обращения: 2024-05-02)

³⁸<https://docs.djangoproject.com/en/5.0/ref/contrib/admin/> (дата обращения: 2024-05-02)



Рис. 11: Пример визуализации NDVI в приложении

Пример заполнения данных севооборота и отображения урожайности в приложении представлен на рис. 12.

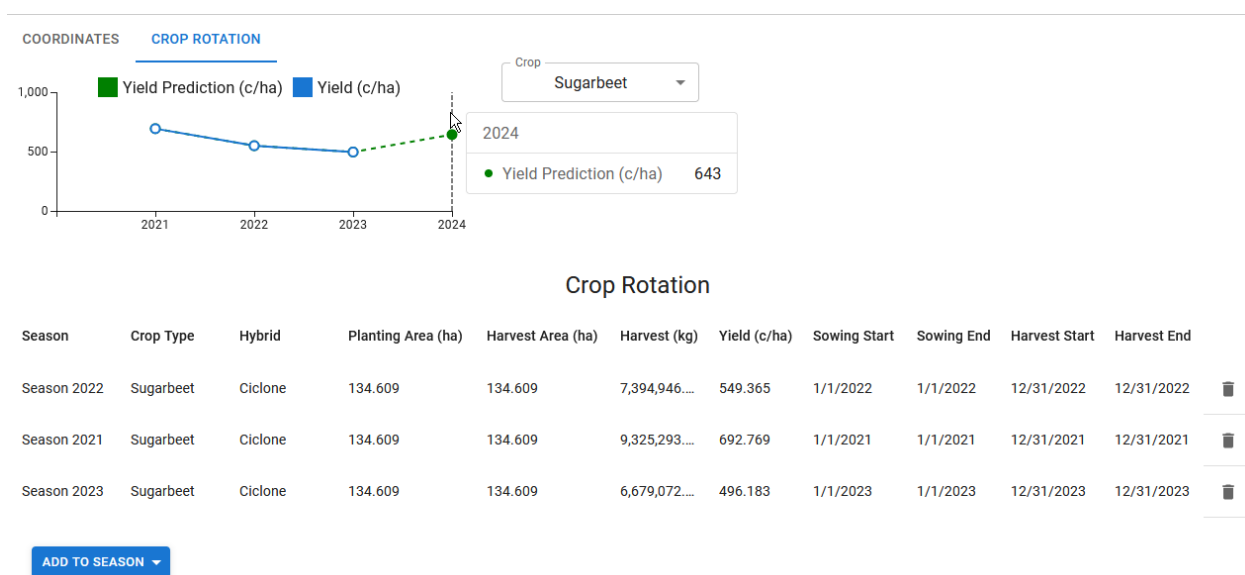


Рис. 12: Пример заполнения данных севооборота и отображения урожайности в приложении

4.6. Локализация интерфейса

Для того чтобы приложение было удобно в работе как для русскоговорящих, так и не русскоговорящих пользователей, его интерфейс доступен на русском и английском языках. Механизм локализации интерфейса приложения реализован при помощи библиотеки `i18next` [16]. Она реализует механизм внедрения локализованных строк в компоненты приложения, а также функции интерполяции строк, обработки множественного числа, форматирования строк и выбора языка интерфейса на основе настроек браузера пользователя. Помимо локализации языка интерфейса, в используемых приложением компонентах библиотеки `Material UI` [24] реализован механизм локализации формата даты.

4.7. Тестирование

Для проверки корректности работы отдельных компонентов приложения и фиксации изменений код был покрыт юнит-тестами.

При написании тестов на серверную часть приложения использовалась библиотека `pytest` [62], предоставляющая функции для проверки утверждений, параметризации тестов, а также механизм их запуска. Вместе с тем был использован плагин `pytest-django`³⁹, упрощающий тестирование Django-приложения за счет управления его конфигурацией, функций для тестирования с использованием базы данных и выполнения тестовых запросов.

Юнит-тесты для клиентской части приложения написаны при помощи библиотеки `jest` [19]. Она предоставляет широкий набор функций для создания тестового окружения для клиента, включая API сервера и виртуальное дерево компонентов. Для тестирования кода приложения, отвечающего за управление состоянием, используется часть библиотеки `Redux` `redux-mock-store`⁴⁰.

³⁹<https://pytest-django.readthedocs.io/en/latest/> (дата обращения: 2024-05-02)

⁴⁰<https://github.com/reduxjs/redux-mock-store> (дата обращения: 2024-05-02)

4.8. Непрерывное развертывание

Монолитная архитектура приложения значительно упрощает его развертывание, так как не нужно развертывать разные сервисы отдельно и настраивать взаимодействие между ними. Независимость от окружения, версионирование зависимостей и изоляция от других запущенных приложений обеспечивается его развертыванием в виде Docker-контейнера [8]. Для развертывания приложения была выбрана облачная платформа Yandex Cloud [63], так как Skoltech Agro использует ее в других проектах. Для своей работы приложение использует следующие сервисы платформы:

- Managed Service for PostgreSQL⁴¹ для хранения данных;
- Yandex Container Registry⁴² для хранения и версионирования Docker-образа;
- Yandex Serverless Containers⁴³ для управления запуском контейнеров;
- Yandex Lockbox⁴⁴ для хранения секретов.

Алгоритм публикации новой версии приложения реализован через GitHub Actions [11]. После внесения изменений в основную ветку приложения на основе файлов исходного кода собирается новая версия Docker-образа, которая затем загружается в регистр контейнеров Yandex Container Registry, затем новая версия образа передается в сервис Yandex Serverless Containers для запуска контейнеров, после чего новая версия приложения становится доступной пользователям.

⁴¹<https://yandex.cloud/en-ru/services/managed-postgresql> (дата обращения: 2024-05-02)

⁴²<https://yandex.cloud/en-ru/services/container-registry> (дата обращения: 2024-05-02)

⁴³<https://yandex.cloud/en-ru/services/serverless-containers> (дата обращения: 2024-05-02)

⁴⁴<https://yandex.cloud/en-ru/services/lockbox> (дата обращения: 2024-05-02)

5. Апробация

Для того чтобы получить оценку удобства пользовательского интерфейса, была проведена апробация с использованием методики System Usability Scale [50]. Эта методика позволяет получить числовую оценку качества пользовательского интерфейса, что позволяет отслеживать изменения этого показателя и сравнивать разные версии приложения между собой. Помимо этого, ее использование позволяет собрать развернутые отзывы пользователей, чтобы локализовать недостатки системы.

В рамках апробации восьми студентам и сотрудникам Skoltech Agro было предложено ознакомиться с инструкцией по работе с приложением⁴⁵, а затем выполнить ряд заданий, используя приложение. Предложенные задания покрывали основную функциональность приложения, включая создание пользовательских аккаунтов, работу с картой, загрузку данных о фермерском хозяйстве и севообороте, изучение показателей характеристик посевов и прогноза урожайности. Согласно методике, после выполнения заданий, пользователям было предложено ответить на 10 вопросов, касающихся удобства работы с системой. После получения всех ответов, по формуле была посчитана итоговая оценка равная 82,5, что говорит о том, что пользоваться приложением удобно. Анализ ответов показал, что ключевыми трудностями, с которыми сталкивались участники исследования стали:

- недостаток функциональности для работы с данными в разных сезонах,
- отсутствие кеширования результатов вычисления показателей характеристик посевов,
- отсутствие возможности сравнить состояние посевов на карте в две разные даты.

.

⁴⁵<https://easy-agro.notion.site/9e5a25a1393542e087fa4573d2c92d5f?v=8c66d9bff5974903bf2a8cd35a466e4d> (дата обращения: 2024-05-02)

Данные недостатки могут быть исправлены в следующих версиях приложения. Кеширование результатов вычисления показателей характеристик посевов также позволит сократить количество обращений к API Google Earth Engine.

Заключение

В результате работы в были решены следующие задачи.

- Выполнен обзор существующих аналогичных решений.
- Сформулированы требования к приложению.
- Выполнен обзор технологий для создания веб-приложения, реализующего функциональность из требований.
- Разработана архитектура приложения.
- Реализовано приложение согласно требованиям и архитектуре.
- Решение было покрыто тестами.
- Реализовано непрерывное развертывание приложения.
- Проведена апробация решения.

Исходный код проекта закрыт, находится под соглашением о неразглашении.

Список литературы

- [1] 8 - Hyperspectral remote sensing in precision agriculture: present status, challenges, and future trends / Prachi Singh, Prem Chandra Pandey, George P. Petropoulos et al. — 2020. — P. 121–146.
- [2] AQUASTAT - FAO's Global Information System on Water and Agriculture. — URL: <https://www.fao.org/aquastat/ru/overview/methodology/water-use> (дата обращения: 2024-05-02).
- [3] ASP.NET Core. — URL: <https://dotnet.microsoft.com/en-us/apps/aspnet> (дата обращения: 2024-05-02).
- [4] Angular. — URL: <https://angular.io/> (дата обращения: 2024-05-02).
- [5] Bing Maps API. — URL: <https://www.microsoft.com/en-us/maps/bing-maps/> (дата обращения: 2024-05-02).
- [6] Cross-site request forgery. — URL: https://en.wikipedia.org/wiki/Cross-site_request_forgery (дата обращения: 2024-05-02).
- [7] Django web framework. — URL: <https://www.djangoproject.com> (дата обращения: 2024-05-02).
- [8] Docker: Accelerated Container Application Development. — URL: <https://www.docker.com/> (дата обращения: 2024-05-02).
- [9] Fielding Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures // Doctoral dissertation, University of California. — 2000. — URL: <https://ics.uci.edu/~fielding/pubs/dissertation/top.htm>.
- [10] Flask documentation. — URL: <https://flask.palletsprojects.com/en/3.0.x/> (дата обращения: 2024-05-02).
- [11] GitHub Actions. — URL: <https://github.com/features/actions> (дата обращения: 2024-05-02).

- [12] Global Interpreter Lock. — URL: <https://wiki.python.org/moin/GlobalInterpreterLock> (дата обращения: 2024-05-02).
- [13] Google Maps Platform. — URL: <https://mapsplatform.google.com/> (дата обращения: 2024-05-02).
- [14] Gunicorn - Python WSGI HTTP server for UNIX. — URL: <https://gunicorn.org/> (дата обращения: 2024-05-02).
- [15] Integrating a Modern JavaScript Pipeline into a Django Application. — URL: <https://www.saaspegasus.com/guides/modern-javascript-for-django-developers/integrating-javascript-pipeline/> (дата обращения: 2024-05-02).
- [16] Introduction | i18next documentation. — URL: <https://www.i18next.com/> (дата обращения: 2024-05-02).
- [17] JS framework benchmark. — URL: <https://krausest.github.io/js-framework-benchmark/current.html> (дата обращения: 2024-05-02).
- [18] JSON Web Token. — URL: https://en.wikipedia.org/wiki/JSON_Web_Token (дата обращения: 2024-05-02).
- [19] Jest | Delightful JavaScript Testing. — URL: <https://jestjs.io/> (дата обращения: 2024-05-02).
- [20] Keyhole Markup Language. — URL: https://en.wikipedia.org/wiki/Keyhole_Markup_Language (дата обращения: 2024-05-02).
- [21] Leaflet - a JavaScript library for interactive maps. — URL: <https://leafletjs.com/> (дата обращения: 2024-05-02).
- [22] Long-term impact of a precision agriculture system on grain crop production / M. A. Yost, N. R. Kitchen, K. A. Sudduth et al. // [Precision Agriculture](#). — 2017. — 10. — Vol. 18.

- [23] Mapbox | Maps, Navigation, Search and Data. — URL: <https://www.mapbox.com/> (дата обращения: 2024-05-02).
- [24] Material UI: React components that implement Material Design. — URL: <https://mui.com/material-ui/> (дата обращения: 2024-05-02).
- [25] Microservices vs. monolithic architecture. — URL: <https://www.atlassian.com/ru/microservices/microservices-architecture/microservices-vs-monolith> (дата обращения: 2024-05-02).
- [26] The Most Popular Front-end Frameworks. — URL: <https://stackdiary.com/front-end-frameworks/> (дата обращения: 2024-05-02).
- [27] MySQL. — URL: <https://www.mysql.com/> (дата обращения: 2024-05-02).
- [28] NDMI: Нормализованный Разностный Индекс Влажности. — URL: <https://eos.com/ru/make-an-analysis/ndmi/> (дата обращения: 2024-05-02).
- [29] Node.js. — URL: <https://nodejs.org/en> (дата обращения: 2024-05-02).
- [30] Nominatim. — URL: <https://nominatim.org/release-docs/latest/> (дата обращения: 2024-05-02).
- [31] OneSoil | Free Farming App for Precision Agriculture. — URL: <https://onesoil.ai/en> (дата обращения: 2024-05-02).
- [32] OpenLayers. — URL: <https://openlayers.org/> (дата обращения: 2024-05-02).
- [33] OpenLayers vs Leaflet! Performance and Functionality Comparison. — URL: <https://blog.teoman.me/openlayers-vs-leaflet-which-one-should-we-use/> (дата обращения: 2024-05-02).

- [34] PEP 3333 – Python Web Server Gateway Interface v1.0.1. — URL: <https://peps.python.org/pep-3333/> (дата обращения: 2024-05-02).
- [35] PostGIS. — URL: <https://postgis.net/> (дата обращения: 2024-05-02).
- [36] PostgreSQL: The World’s Most Advanced Open Source Relational Database. — URL: <https://www.postgresql.org/> (дата обращения: 2024-05-02).
- [37] PostgreSQL database as a service | Yandex Cloud - services. — URL: <https://yandex.cloud/en-ru/services/managed-postgresql> (дата обращения: 2024-05-02).
- [38] The Power of Drone and Satellite Imagery in Agriculture. — URL: <https://www.cloudeo.group/blog/cloudeo-blog-space-1/the-power-of-drone-and-satellite-imagery-in-agriculture-112> (дата обращения: 2024-05-02).
- [39] React. — URL: <https://react.dev/> (дата обращения: 2024-05-02).
- [40] React Native. — URL: <https://reactnative.dev/> (дата обращения: 2024-05-02).
- [41] React Redux | React Redux. — URL: <https://react-redux.js.org/> (дата обращения: 2024-05-02).
- [42] Redux - A JS library for predictable and maintainable global state management. — URL: <https://redux.js.org/> (дата обращения: 2024-05-02).
- [43] Redux Fundamentals. — URL: <https://redux.js.org/tutorials/fundamentals/part-2-concepts-data-flow> (дата обращения: 2024-05-02).
- [44] Redux Toolkit | Redux Toolkit. — URL: <https://redux-toolkit.js.org/> (дата обращения: 2024-05-02).

- [45] S Shyam Mohan J, Goswami Krishanu. Performance Analysis and Comparison of Node.js and Java Spring Boot in Implementation of Restful Applications. — 2023. — 10. — URL: <https://www.authorea.com/users/671723/articles/671117-performance-analysis-and-comparison-of-node-js-and-java->
- [46] SQLite Home Page. — URL: <https://www.sqlite.org/index.html> (дата обращения: 2024-05-02).
- [47] Single Page Application. — URL: https://en.wikipedia.org/wiki/Single-page_application (дата обращения: 2024-05-02).
- [48] Spring boot. — URL: <https://spring.io/projects/spring-boot> (дата обращения: 2024-05-02).
- [49] Stefanakis Emmanuel. Web mercator and raster tile maps: two cornerstones of online map service providers // [GEOMATICA](#). — 2017. — 06. — Vol. 71. — P. 100–109.
- [50] System Usability Scale for Data-Driven UX. — URL: <https://www.interaction-design.org/literature/article/system-usability-scale> (дата обращения: 2024-05-02).
- [51] TechEmpower Framework Benchmarks. — URL: <https://www.techempower.com/benchmarks/> (дата обращения: 2024-05-02).
- [52] TypeScript Roadmap: January–June 2019. — URL: <https://github.com/Microsoft/TypeScript/issues/29288> (дата обращения: 2024-05-02).
- [53] TypeScript is JavaScript with syntax for types. — URL: <https://www.typescriptlang.org/> (дата обращения: 2024-05-02).
- [54] User Guides - Sentinel-2 MSI - Level 2 Processing. — URL: <https://sentinel.esa.int/web/sentinel/user-guides/sentinel-2-msi/processing-levels/level-2> (дата обращения: 2024-05-02).

- [55] User Guides - Sentinel-2 MSI - Level 2 Processing. — URL: <https://sentinel.esa.int/web/sentinel/user-guides/sentinel-2-msi/processing-levels/level-2> (дата обращения: 2024-05-02).
- [56] Vue.js - The Progressive JavaScript Framework. — URL: <https://vuejs.org/> (дата обращения: 2024-05-02).
- [57] What is SaaS? Software as a service | Microsoft Azure. — URL: <https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-saas> (дата обращения: 2024-05-02).
- [58] Why not use Python for everything (like enterprise apps)? — URL: <https://themastermind.medium.com/why-not-use-python-for-everything-like-enterprise-apps-24fcccd7> (дата обращения: 2024-05-02).
- [59] djoser. — URL: <https://djoser.readthedocs.io/en/latest/introduction.html> (дата обращения: 2024-05-02).
- [60] maplibre-gl-js. — URL: <https://github.com/maplibre/maplibre-gl-js> (дата обращения: 2024-05-02).
- [61] nginx. — URL: <https://nginx.org/ru/> (дата обращения: 2024-05-02).
- [62] pytest documentation. — URL: <https://docs.pytest.org/en/8.1.x/> (дата обращения: 2024-05-02).
- [63] The secure, fault-tolerant cloud platform from Yandex | Yandex Cloud. — URL: <https://yandex.cloud/en-ru/> (дата обращения: 2024-05-02).
- [64] Основные итоги сельскохозяйственной микропереписи 2021 года. — URL: https://rosstat.gov.ru/storage/mediabank/Census_agr_2021_final.pdf (дата обращения: 2024-05-02).

- [65] Платформа цифрового сельского хозяйства | Exact Farming. — URL: <https://exactfarming.com/> (дата обращения: 2024-05-02).
- [66] Что такое индекс NDVI и как он делает жизнь фермера проще. — URL: <https://blog.onesoil.ai/ru/what-is-ndvi> (дата обращения: 2024-05-02).