

About the authors:

GROMOVA Polina Aleksandrovna – Undergraduate Student, Tver State Technical University, Tver. E-mail: pa.gromova@gmail.ru

KRUTSKIKH Elena Igorevna – Senior Lecturer of the Department of Software, Tver State Technical University, Tver. E-mail: yelena.korneeva@ya.ru

УДК 336.226.212.1

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ УПРАВЛЕНИЯ САДОВЫМ УЧАСТКОМ С ИСПОЛЬЗОВАНИЕМ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ И МЕТОДОВ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Т.В. Латыпов, Е.И. Крутских

© Латыпов Т.В., Крутских Е.И., 2026

***Аннотация.** Статья посвящена разработке программного комплекса для автоматизации процессов садоводства. Рассмотрена архитектура системы, включающая мобильное приложение, серверную часть на основе микросервисов и сервис садового ассистента. Описаны процессы сбора и обработки данных о растениях с применением веб-скреппинга и языковых моделей, а также реализация функций распознавания растений по фотографии упаковки семян. Приведены результаты проектирования и реализации основных модулей системы.*

***Ключевые слова:** садоводство, мобильное приложение, микросервисная архитектура, искусственный интеллект, машинное обучение, распознавание образов, веб-скреппинг, Flutter, ASP.NET Core, FastAPI.*

Введение

Современное садоводство, как любительское, так и профессиональное, связано с необходимостью учета большого количества данных: видов растений, графика работ, планов участка и т. д. Существующие на рынке приложения зачастую характеризуются ограниченным функционалом в бесплатных версиях, низким качеством распознавания, отсутствием локализации или недостаточной автоматизацией процессов [1, 2]. Целью данной работы является создание комплексного мобильного приложения, не имеющего указанных недостатков и предоставляющего пользователю удобный инструментарий для полного цикла управления садовым участком.

Архитектура системы

Программный комплекс построен на основе микросервисной архитектуры, что обеспечивает масштабируемость, отказоустойчивость и легкость поддержки [3]. Ключевыми компонентами системы являются:

1. Мобильное приложение. Разработано на фреймворке Flutter (язык Dart) [4], что обеспечивает кросс-платформенность для iOS и Android. Для офлайн-работы используется локальная база данных NoSQL Hive.

2. Серверная часть. Реализована на основе ASP.NET Core (C#) [5] и включает несколько микросервисов:

AuthService – управление аутентификацией и авторизацией с использованием JWT-токенов;

GardenService – учет растений и садовых работ (CRUD-операции);

AssistantService – управление виртуальными садами и формирование расписания работ.

3. Сервис распознавания (Recognizer). Реализован на Python с использованием FastAPI. Выполняет две основные функции:

распознавание текста с фотографии упаковки семян с помощью библиотеки PyTesseract [6];

поиск наиболее релевантного растения в базе данных по распознанному тексту с использованием алгоритма Левенштейна для определения схожести строк.

4. API Gateway. На базе Ocelot обеспечивает единую точку входа, маршрутизацию запросов и фильтрацию неавторизованного доступа.

5. База данных. В качестве основной системы управления базами данных используется PostgreSQL [7]. Каждый микросервис взаимодействует с определенным набором таблиц.

6. Система мониторинга. Для сбора метрик и логов применяется стек Prometheus, Loki и Grafana.

Все компоненты системы контейнеризированы с помощью Docker [8] и оркестрированы через Docker Compose.

Методы и подходы

Одной из ключевых задач являлось формирование обширной базы знаний о растениях. Для ее решения был применен веб-скреппинг с использованием библиотеки Playwright для Python [9]. Были обработаны сайты крупнейших российских поставщиков семян и садоводческие форумы. Полученные данные прошли этап структурирования, форматирования и перевода с использованием Google Translate API.

Для автоматизации описания видов растений была задействована языковая модель Gemma3 4B [10]. Для классификации потребностей растений в поливе и освещении по текстовым описаниям была обучена модель машинного обучения с использованием библиотеки scikit-learn на предварительно размеченном вручную датасете [11].

Функция распознавания растения по упаковке реализована следующим образом: пользователь загружает фотографию, сервис Recognizer с помощью PyTesseract извлекает текст, который затем сопоставляется с базой данных по названию и описанию. Для поиска наилучшего соответствия используется расстояние Левенштейна.

Результаты и обсуждение

Созданный полнофункциональный программный комплекс включает в себя:

1. Мобильное приложение с интерфейсами:

для учета растений (создания, редактирования, группировки списков растений с возможностью указания их научных названий, токсичности, периода созревания и привязки к группе);

календаря работ (визуализации графика полива, подкормки и других мероприятий в виде дня, недели или месяца);

визуализации сада (создания плана участка «сверху» с размещением растений).

2. Серверную часть, обрабатывающую запросы от мобильного приложения и обеспечивающую целостность данных.

3. Сервис-ассистент, который значительно упрощает добавление новых растений в коллекцию пользователя за счет автоматического распознавания и предзаполнения данных.

Проведенное юнит- и интеграционное тестирование с помощью NUnit, а также сквозное тестирование с использованием стэка мониторинга подтвердило корректность работы всех модулей системы.

Результаты сравнительного анализа функциональных возможностей разработанного и существующих приложений Plantify и «Садовый менеджер» приведены ниже:

Функциональность	Разработанное приложение	Plantify	«Садовый менеджер»
Учет растений	+	+	+
Календарь работ	+	+	± (ограничено)
Планирование участка	+	—	—
Распознавание по фото упаковки	+	—	—

Функциональность	Разработанное приложение	Plantify	«Садовый менеджер»
Локализация (русский язык)	+	–	+
Автоматическое предзаполнение	+	–	–
Офлайн-режим	+	–	–

Заключение

В рамках проведенного исследования была успешно разработана и протестирована комплексная система для управления садовым участком. Применение микросервисной архитектуры и современных технологий, таких как Flutter, ASP.NET Core, FastAPI и машинное обучение, позволило создать масштабируемое, производительное и удобное для пользователя решение.

К перспективным направлениям дальнейшего развития проекта можно отнести внедрение модели распознавания растения по фотографии самого растения (а не упаковки), создание автоматизированной системы корректировки расписания ухода за растениями на основе прогноза погоды и информации, полученной с датчиков окружающей среды, а также углубленный сбор и анализ статистики для построения прогнозных моделей роста и урожайности.

Библиографический список

1. Plantify: Plant Identifier [Мобильное приложение] // Google Play. URL: <https://play.google.com/store/apps/details?id=com.plantify.app> (дата обращения: 14.11.2025).
2. Садовый менеджер [Мобильное приложение] // Google Play. URL: <https://play.google.com/store/apps/details?id=ru.sadovod.gardenmanager> (дата обращения: 14.11.2025).
3. Ньюмен С. Создание микросервисов. 2-е изд. СПб.: Питер, 2023. 624 с.
4. Flutter Documentation. URL: <https://docs.flutter.dev/> (дата обращения: 26.12.2025).
5. Microsoft ASP.NET Overview. URL: <https://learn.microsoft.com/ru-ru/aspnet/overview> (дата обращения: 26.12.2025).
6. Tesseract OCR Documentation. URL: <https://tesseract-ocr.github.io/tessdoc/#tesseract-user-manual> (дата обращения: 26.12.2025).

7. PostgreSQL Documentation. URL: <https://www.postgresql.org/docs/current/> (дата обращения: 26.12.2025).
8. Docker Documentation. URL: <https://docs.docker.com/> (дата обращения: 26.12.2025).
9. Playwright Documentation. URL: <https://playwright.dev/docs/intro> (дата обращения: 21.01.2026).
10. Gemma: Quantized AI for Consumer GPUs // Google Developers Blog. URL: <https://developers.googleblog.com/en/gemma-3-quantized-aware-trained-state-of-the-art-ai-to-consumer-gpus/> (дата обращения: 21.01.2026).
11. Microsoft: Training Classification Models. URL: <https://learn.microsoft.com/ru-ru/training/modules/train-evaluate-classification-models/> (дата обращения: 21.01.2026).

DEVELOPMENT OF A MOBILE APPLICATION FOR GARDEN MANAGEMENT USING MICROSERVICE ARCHITECTURE AND ARTIFICIAL INTELLIGENCE METHODS

T.V. Latypov, E.I. Krutskikh

Abstract. *The article is devoted to the development of a software complex for automation of gardening processes. The system architecture is considered, including a mobile application, a microservice-based server part and a garden assistant service. The processes of collecting and processing plant data using web scraping and language models, as well as the implementation of plant recognition functions from a photo of a seed package, are described. The results of designing and implementing the main modules of the system are presented.*

Keywords: *gardening, mobile application, microservice architecture, artificial intelligence, machine learning, pattern recognition, web scraping, Flutter, ASP.NET Core, FastAPI.*

Об авторах:

ЛАТЫПОВ Тимофей Вадимович – магистрант, ФГБОУ ВО «Тверской государственный технический университет», Тверь. E-mail: timtompson3241@yandex.ru

КРУТСКИХ Елена Игоревна – старший преподаватель кафедры программного обеспечения, ФГБОУ ВО «Тверской государственный технический университет», Тверь. E-mail: yelena.korneeva@ya.ru

About the authors:

LATYPOV Timofey Vadimovich – Undergraduate Student, Tver State Technical University, Tver. E-mail: timtompson3241@yandex.ru

KRUTSKIKH Elena Igorevna – Senior Lecturer of the Department of Software, Tver State Technical University, Tver. E-mail: yelena.korneeva@ya.ru

УДК 681.5.01

ОСНОВЫ ТЕХНОЛОГИИ ПЛАТФОРМЕННОГО ВЫЗОВА P/INVOKE В ПРОГРАММНОЙ ПЛАТФОРМЕ .NET И ЯЗЫКЕ ПРОГРАММИРОВАНИЯ C#

И.Т. Новиков, В.В. Лебедев

© Новиков И.Т., Лебедев В.В., 2026

***Аннотация.** В статье рассмотрены основные понятия и возможности технологии P/Invoke, предназначенной для вызова нативных (неуправляемых) функций внутри управляемой исполняющей среды CLR программной платформы .NET.*

***Ключевые слова:** C#, .NET, P/Invoke, неуправляемые функции, нативный код, маршалинг.*

Приложения на языке программирования C# разрабатываются на базе программной платформы .NET (бывшая .NET Core, предыдущая версия – .NET Framework). Платформа .NET приносит много удобств в программирование: сборщик мусора, JIT-компиляция, LINQ, большой функционал для языков программирования и многое другое. Однако наряду с преимуществами у .NET есть и минусы. Программы на платформе .NET компилируются в промежуточный язык программирования (CIL), который транслируется в машинный код с помощью виртуальной машины (CLR) [1]. Это сильно снижает производительность приложений, хотя и дает возможность реализовать весь перечень удобств, предоставляемый платформой. Тем не менее иногда программисту важна именно производительность, а в некоторых случаях у разработчика уже есть библиотека на быстром, «нативном» языке программирования, таком как C или C++. Возможно ли разработать приложение на языке C# в рамках платформы .NET с использованием уже ранее написанных на нативных языках библиотек, представленных машинным кодом, а не промежуточным языком CIL, который используется в .NET? На этот вопрос можно ответить положительно, так как платформа .NET предоставляет целый перечень возможностей для взаимодействия с нативными языками программирования. В этой статье мы рассмотрим такой механизм взаимодействия .NET и нативных языков, как P/Invoke.