ISBN 978-5-901722-96-1

УДК 004.45

РАЗРАБОТКА TELEGRAM-ПЛАТФОРМЫ ДЛЯ ФОРМИРОВАНИЯ КОМАНД УЧАСТНИКОВ ХАКАТОНОВ

Синичкина Милана Дмитриевна[1], бакалавр
e-mail: plmrv2000@gmail.ru
Тимофеев Данил Николаевич[1], бакалавр
e-mail: timofeevdanil28@gmail.ru
Шакирзянов Ринат Михайлович[1], кандидат технических наук, доцент кафедры ПМИ
e-mail: rmshakirzyanov@kai.ru

[1] Казанский национальный исследовательский технический университет имени А.Н. Туполева, Казань, Россия

Аннотация. В статье представлена разработка платформы для поиска участников команды на хакатоны (Telegram-бот + Mini App). Решение решает проблему потери информации о навыках и ролях в общих чатах, ускоряя подбор тиммейтов: профиль, выбор мероприятия, просмотр участников, взаимный интерес. Описаны требования, архитектура, алгоритм мэтчинга, хранение данных и результаты тестирования HackathonBot.

Ключевые слова: хакатон, Telegram-бот, Telegram Mini App, поиск команды, мэтчинг участников, FastAPI, React, PostgreSQL.

DEVELOPMENT OF A TELEGRAM PLATFORM FOR FORMING TEAMS OF HACKATHON PARTICIPANTS

Milana D. Sinichkina[1], Bachelor’s Degree Student
e-mail: plmrv2000@gmail.ru
Danil N. Timofeev[1], Bachelor’s Degree Student
e-mail: timofeevdanil28@gmail.ru
Rinat M. Shakirzyanov[1], Candidate of Technical Sciences, Associate Professor of the Department of PMI
e-mail: rmshakirzyanov@kai.ru

[1] Kazan National Research Technical University named after A. N. Tupolev, Kazan, Russia

Abstract. The article presents the development of a platform for finding team members for hackathons (Telegram bot + Mini App). The solution solves the problem of losing information about skills and roles in shared chats, speeding up the selection of teammates: profile, event selection, viewing participants, mutual interest. The requirements, architecture, matching algorithm, data storage, and HackathonBot test results are described.

Keywords: hackathon, Telegram bot, Telegram Mini App, team search, participant matching, FastAPI, React, PostgreSQL.

Введение

Хакатоны стали распространённым форматом практической разработки, в котором участникам необходимо за короткое время перейти от идеи к работающему прототипу. Для такого формата важна не только сама тема задания, но и состав команды. Если в команде отсутствует backend-разработчик, frontend-разработчик, дизайнер, аналитик или участник, отвечающий за презентацию, работа может замедлиться уже на начальном этапе. Поэтому задача поиска тиммейтов является практической частью подготовки к хакатону, а не второстепенным организационным вопросом.

На практике поиск команды часто выполняется через общие Telegram-чаты, объявления организаторов или личные знакомства. Такой способ остаётся привычным, но имеет ряд ограничений. Сообщения быстро уходят вверх, участники описывают свои навыки в свободной форме, а поиск по чату не всегда помогает найти человека с нужной ролью. Кроме того, один чат может использоваться сразу для нескольких мероприятий, из-за чего теряется контекст конкретного хакатона. В результате участник тратит время не на подготовку идеи и технического решения, а на повторное размещение объявлений и ручное сравнение профилей.

Целью разработки стало создание платформы для поиска команды для хакатонов на основе Telegram-бота и Telegram Mini App. Выбор Telegram как среды работы связан с тем, что многие IT-сообщества и организаторы мероприятий уже используют этот мессенджер для публикации новостей, общения и рассылки уведомлений. Telegram Mini Apps позволяют встроить веб-интерфейс непосредственно в мессенджер, а Bot API обеспечивает команды, уведомления и обработку сообщений [1, 2]. Поэтому пользователь не должен переходить на отдельный сайт и проходить самостоятельную регистрацию через логин и пароль.

Проектирование платформы для поиска команды

Разработанная платформа HackathonBot ориентирована на участника хакатона. Пользователь открывает Mini App из бота, проходит идентификацию через данные Telegram Web App, заполняет профиль, выбирает интересующее мероприятие и просматривает карточки других участников. Если пользователь проявляет интерес к другому участнику, система фиксирует лайк. При взаимном лайке создаётся мэтч, после чего участники могут перейти к общению. Такая механика не формирует полноценную команду автоматически, но помогает решить первый и наиболее сложный этап - установить контакт между людьми, которым потенциально интересно работать вместе.

При проектировании были выделены функциональные и нефункциональные требования. К функциональным требованиям относятся идентификация пользователя через Telegram Mini App, ведение профиля, просмотр и фильтрация хакатонов, регистрация на мероприятие, просмотр участников, постановка лайков и дизлайков, создание мэтча при взаимном лайке, обмен сообщениями, перевод сообщений и административное управление каталогом хакатонов. Нефункциональные требования связаны с работой внутри Telegram, хранением данных в PostgreSQL, развёртыванием через Docker Compose, использованием

Идентификация пользователя построена без классической регистрации. На клиентской стороне Mini App получает строку initData из объекта window.Telegram.WebApp и добавляет её в заголовок X-Init-Data при запросах к API. На сервере выполняется проверка этих данных, после чего пользователь ищется в базе по Telegram ID. Если запись отсутствует, создаётся новый профиль с базовыми значениями. Такой подход соответствует

логике Telegram Mini Apps и снижает количество действий, которые должен выполнить пользователь перед началом работы.

Структура данных системы построена вокруг сущностей пользователя, хакатона, участия пользователя в хакатоне, лайка, дизлайка, мэтча, сообщения и тематической подписки. Таблица users хранит Telegram ID, имя, username, роль, навыки, языковые настройки, ссылку на GitHub, описание, проекты, признаки активности и администратора. Таблица hackathons содержит сведения о мероприятии: название, описание, даты, темы, ссылку, формат, локацию, размер команды и контакты. Связь между пользователями и хакатонами вынесена в таблицу user_hackathons, поскольку один пользователь может участвовать в нескольких мероприятиях, а одно мероприятие объединяет много участников.

Алгоритм подбора участников основан на ленте кандидатов и действиях пользователя. При получении рекомендаций система исключает самого текущего пользователя, а также тех участников, которым он уже поставил лайк или дизлайк. Если выбран конкретный хакатон, кандидаты дополнительно ограничиваются участниками, зарегистрированными на это мероприятие. Это позволяет не смешивать разные события и не показывать пользователю одни и те же карточки повторно.

Обработка лайка включает несколько проверок. Сначала сервер убеждается, что оба пользователя существуют. Если действие выполняется в контексте хакатона, проверяется участие обоих пользователей в этом мероприятии. Затем удаляется возможный ранее поставленный дизлайк, потому что последнее действие пользователя должно считаться актуальным. После сохранения лайка система ищет встречный лайк от второго пользователя. Если он найден, создаётся запись в таблице matches, где фиксируется пара пользователей и контекст хакатона. При необходимости бот может отправить участникам уведомление о совпадении. Последовательность обработки лайка и создания мэтча при взаимном интересе пользователей представлена на рисунке 1.

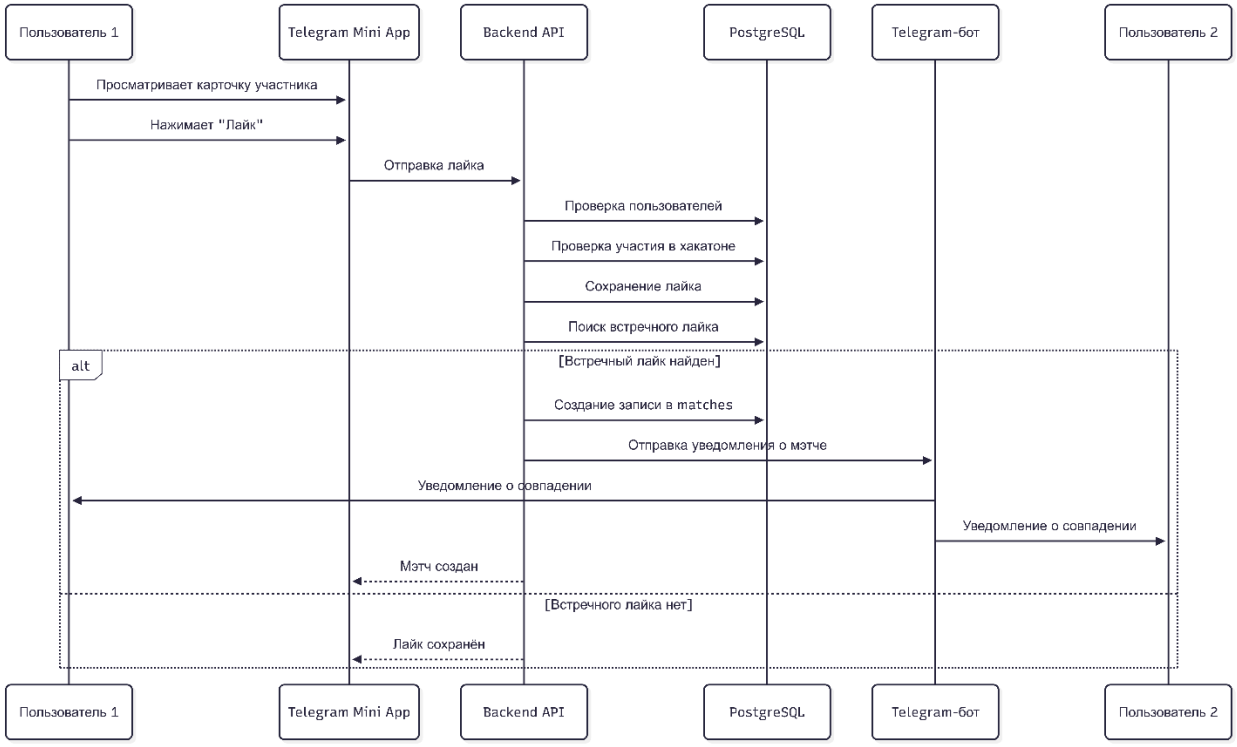

Рисунок 1 — Алгоритм создания мэтча при взаимном лайке

Реализация и результаты работы системы

Архитектура платформы включает несколько связанных компонентов. Серверная часть реализована на FastAPI, что удобно для построения REST API с типизированными схемами и автоматической документацией [3]. Telegram-бот разработан с использованием aiogram, клиентская часть выполнена на React, TypeScript и Vite, а стилизация интерфейса построена на Tailwind CSS. Для хранения данных используется PostgreSQL, вспомогательное кэширование выполняется через Redis, а перевод сообщений реализован через локально развёрнутый LibreTranslate. Развёртывание всех компонентов организовано в Docker Compose, что упрощает запуск проекта в единой среде [4]. Общая структура разработанной платформы представлена на рисунке 2.

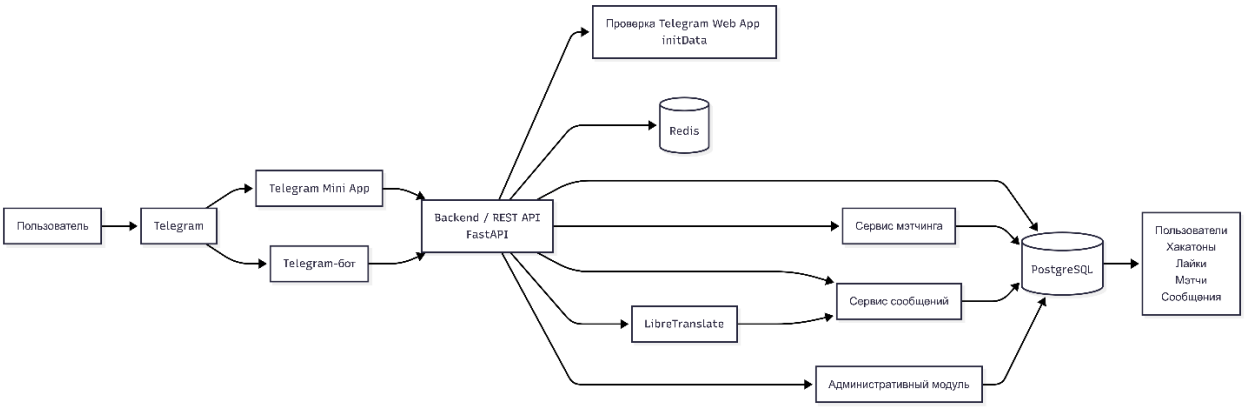

Рисунок 2 — Общая архитектура платформы HackathonBot

Серверная часть разделена по предметным модулям. Модуль авторизации отвечает за получение или создание пользователя на основе Telegram Web App init data. Модуль пользователей хранит профиль участника, включая роль, навыки, язык общения, ссылку на GitHub, описание и проекты. Модуль хакатонов предоставляет каталог мероприятий и регистрацию пользователя на выбранный хакатон. Модуль мэтчинга обрабатывает лайки, дизлайки и совпадения. Отдельно реализованы чаты, административная панель, запуск парсера хакатонов и подписки на тематические уведомления.

После создания мэтча участники могут перейти к общению. Сообщения сохраняются в базе данных с исходным текстом и, если требуется, переведённой версией. Перевод выполняется через LibreTranslate, который запускается локально и не требует передачи переписки во внешние коммерческие сервисы. Эта функция не является основной для самого поиска команды, однако она полезна для онлайн-хакатонов и мероприятий с международным составом участников, где языковой барьер может помешать быстрому распределению ролей.

Клиентская часть реализует основные пользовательские страницы: профиль, каталог хакатонов, поиск участников, команду, настройки и административную панель. На странице профиля пользователь заполняет данные, которые будут видны другим участникам. В каталоге хакатонов он выбирает мероприятие и регистрируется на него. На странице поиска отображаются карточки пользователей, по которым можно поставить лайк или дизлайк. В разделе команды показываются совпадения и входящие или исходящие проявления интереса. Такой набор страниц покрывает основной сценарий от входа в Mini App до начала общения с потенциальным тиммейтом.

Административная часть нужна для поддержки каталога мероприятий и управления пользователями. Администратор может создавать, изменять и удалять хакатоны, менять активность пользователей, просматривать статистику и запускать парсер внешних

источников. Парсер предназначен для загрузки сведений о хакатонах со страниц, после чего система может сохранить новые мероприятия в базу и уведомить пользователей, подписанных на соответствующие темы. При этом ручное управление остаётся необходимым, поскольку внешние страницы могут иметь разную структуру и не всегда содержать полные данные.

Тестирование разработанного приложения проводилось автоматизированно и вручную. Backend-тесты проверяют авторизацию, получение хакатонов, пользователей, лайков и мэтчей. Frontend-тесты охватывают карточки пользователей и хакатонов, модальные окна, загрузку ленты, фильтрацию и отображение уведомлений. Ручная проверка выполнялась в Telegram и включала запуск Mini App, заполнение профиля, регистрацию на хакатон, просмотр участников, создание взаимного лайка и переход к общению.

Полученный результат показывает, что платформа решает поставленную задачу: участник может найти потенциального тиммейта в привычной Telegram-среде, не просматривая вручную длинные переписки. Сильной стороной решения является сочетание бота и Mini App. Бот удобен как точка входа и канал уведомлений, а Mini App предоставляет более структурированный интерфейс для профиля, карточек и списков. Ограничение текущей реализации состоит в том, что команда формируется через парные совпадения. В дальнейшем платформу можно развить до группового подбора с учётом требуемого размера команды, ролей, технологий и статуса участия.

Заключение

Таким образом, разработанная Telegram-платформа может рассматриваться как прикладное средство автоматизации первичного командообразования на хакатонах. Она не заменяет платформы организаторов и системы оценки проектов, но закрывает отдельный пользовательский сценарий: быстрый поиск человека, с которым можно начать обсуждение идеи и распределение задач. Для студентов и начинающих разработчиков такой инструмент особенно полезен, поскольку снижает зависимость от личных контактов и делает процесс поиска команды более прозрачным.

Список литературы:

1. Telegram Mini Apps. Документация Telegram. – URL: https://core.telegram.org/bots/webapps (дата обращения 09.05.2026). – Текст: электронный.

2. Telegram Bot API. Документация Telegram. – URL: https://core.telegram.org/bots/api (дата обращения 09.05.2026). – Текст: электронный.

3. FastAPI Documentation. – URL: https://fastapi.tiangolo.com (дата обращения 09.05.2026). – Текст: электронный.

4. Docker Compose Documentation. – URL: https://docs.docker.com/compose/ (дата обращения 09.05.2026). – Текст: электронный.