АРХИТЕКТУРА ПРИЛОЖЕНИЯ ДЛЯ БЫСТРОГО ОБМЕНА ТЕКСТОВЫМИ СООБЩЕНИЯМИ

APPLICATION ARCHITECTURE FOR FAST EXCHANGE TEXT MESSAGES



УДК 004.02

DOI: 10.24411/2658-4964-2021-10345

Пачкин Игорь Владимирович, студент 4 курс, факультет «Программная инженерия», Институт информационных технологий, Россия, г. Москва

Pachkin Igor Vladimirovich , e-mail altair1901@icloud.com

Аннотация

Цель исследования настоящей работы является рассмотрение архитектуры, нацеленной на приложение для быстрого обмена текстовыми сообщениями. При создании любого приложения, требуется определить несколько параметров, которые существенно повлияют на его разработку и общий жизненный цикл. В этой работе описывается современный подход к разработке такого типа приложения.

Annotation

The purpose of the study this work is to consider an architecture aimed at a fast text messaging application. When creating any application, you need to define several parameters that will significantly affect its development and overall life cycle. This paper describes a modern approach to developing this type of application.

Ключевые слова: мессенджер, клиент-серверные приложения, архитектура ПО, приложение для быстрого обмена сообщениями, децентрализация, хабы.

Keywords: messenger, client-server applications, software architecture, app for fast exchange messages, decentralization, hubs.

Введение

В современном мире невозможно обходится без постоянной коммуникации с людьми, будь то коллеги по работе или семейный чат для обсуждения новостей. Люди стали обмениваться мгновенно доставляемыми сообщениями через мессенджеры.

Мессенджеры – это сложные клиент-серверные приложения, отвечающие за коммуникацию своих пользователей. Такие программы создаются с целью реализовать задачи, которые ставятся их создателями, будь то реализация защищённого обмена сообщениями, система, способная масштабироваться на миллионы одновременных пользователей без существенных задержек во времени, или способах коммуникации сотрудников, который будет доступен внутри закрытого контура организации.

Большинство современных мессенджеров имеют несколько версий клиентских приложений написанных под разные платформы. Так, у популярного мессенджера Telegram есть версии под мобильные устройства под управлением операционных систем iOS и Android. А также Web версия, работающая в современных браузерах, и кроссплатформенное desktop приложение под разные операционные системы.

Серверная же часть зависит от конкретного мессенджера и может отличаться как используемыми транспортными протоколами, так и своей архитектурой. Так, например мессенджер может реализовывать протокол XMPP[1], чтобы быть совместимым с существующими клиентскими приложениями. Или же, как в случае Telegram, который общается по протоколу MTProto[2], мессенджер может реализовывать собственный протокол для удовлетворения необходимых целей. В случае Telegram – это реализация шифрования переписки между пользователями, осуществляемая на уровне протокола.

При создании любого приложения, требуется определить несколько параметров, которые существенно повлияют на его разработку и общий жизненный цикл.

Архитектура

Требуется продумать архитектуру будущего приложения. Правильно выбранная архитектура делает процесс разработки и сопровождения программы более эффективным и простым, позволяет сэкономит очень много времени и финансов.

Программу с хорошей архитектурой проще расширять и изменять, а также тестировать, понимать и отлаживать. По причине того, что сложность написанного кода, как правило, растет гораздо быстрее размеров программы. И если сразу не позаботиться об структурировании и написание чистого кода, то со временем наступает момент, когда ты перестаешь его понимать. И после уже может встать вопрос о жизнеспособности вашего проекта[3].

Вот несколько критериев определяющих хорошую архитектуру:

- Гибкость системы
- Тестируемость системы
- Масштабируемость процесса разработки
- Расширяемость системы
- Эффективность системы

Для данного проекта было выбрано использование модульной архитектуры[4].

Основной задачей при разработке крупных систем можно безоговорочно считать задачу по снижению загруженности. Для реализации этого подхода используется деление на более мелкие логические блоки и модули, каждый из которых, строится из частей меньшего размера, до тех пор, пока самые мелкие части не будут достаточно просты для непосредственного понимания и реализации.

В этом случае программа превращается в конструктор, состоящий из блоков, взаимодействующих между собой по хорошо определенным правилам, что собственно и позволяет контролировать ее сложность.

Децентрализация

Децентрализация подразумевает создание общих узлов обмена данными (хабов). Хабы[5] обрабатывают часть информации о пользователях, что снижение нагрузку с устройства пользователя. На рисунке 1 можно ознакомиться со схемой децентрализации системы.

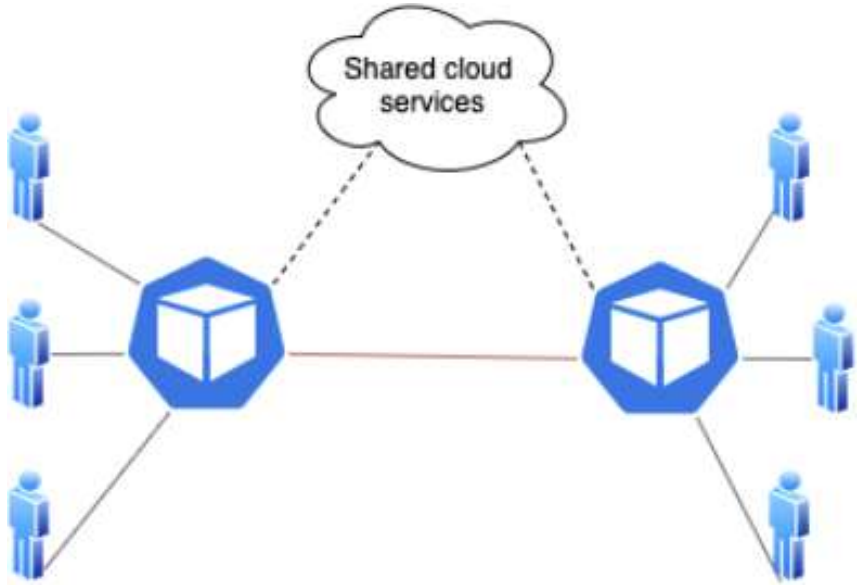


Рисунок 1 – схема децентрализации системы

Как видно из картинки выше, пользователь подключаются к своему узлу. Все узлы равнозначны и образуют одноранговую сеть[6]. Аналогичную сетевую архитектуру образуют сервера электронной почты.

Хабы

В задачи хабов входят:

1. Администрирование пользователей.
2. Резервное копирование данных.
3. Передача данных между пользователями.
4. Хранение файлов.
5. Хранение данных о пользователе.
6. Авторизация, регистрация и верификация пользователей.

Помимо пользователей и хабов, в сети существуют общие микросервисы, к которым обращаются хабы:

Микросервис push-уведомлений. Отвечает за отправку уведомлений на устройства пользователей по распоряжению хаба пользователя.

Микросервис балансировки сети. Обеспечивает целостность сети и является одним из источников информации о работающих в данный момент хабах. Обеспечивает подтверждение хабов и защиту от подделки запросов хаба при взаимодействии между ними и с пользователем.

В случае необходимости, любой хаб может использовать собственные реализации микросервисов.

Заключение

Современные мессенджер должны быть спроектированы с учетом многих факторов, и в процессе создания будущего приложения для быстрого обмена текстовыми сообщениями, рационально опираться на предложенные в контексте данной статьи архитектурные подходы по организации кода системы и их децентрализации.

Литература

1. Wikipedia.org Сводная энциклопедия XMPP [Электронный ресурс] Режим доступа: URL https://ru.wikipedia.org/wiki/XMPP (дата обращения 08.04.2021).
2. Core.telegram.org MTProto Mobile Protocol [Электронный ресурс] Режим доступа: URL https://core.telegram.org/mtproto (дата обращения 08.04.2021).
3. Чистая архитектура. Искусство разработки программного обеспечения: книга / Мартин Роберт — Москва : Питер, 2018. — 352 с.
4. Архитектура программного обеспечения на практике : учебное пособие / Л. Басс, П. Клементс, Р. Кацма — 2-е изд. — Москва : Питер, 2006. — 576 с
5. Wikipedia.org Сводная энциклопедия Хаб [Электронный ресурс] Режим доступа: URL https://ru.wikipedia.org/wiki/%D0%A5%D0%B0%D0%B1 (дата обращения 11.04.2021).

6. Кручинин, В. В. Разработка сетевых приложений : учебное пособие / В. В. Кручинин. — Москва : ТУСУР, 2013. — 120 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/110372 (дата обращения: 11.04.2021). — Режим доступа: для авториз. пользователей.

Literature

1. Wikipedia.org XMPP Consolidated Encyclopedia [Electronic resource] Access mode: URL https://ru.wikipedia.org/wiki/XMPP (date of access 08.04.2021).
2. Mobile protocol Core.telegram.org MTProto [Electronic resource] Access mode: URL https://core.telegram.org/mtproto (date of access 08.04.2021).
3. Clean architecture. The Art of Software Development: Book Robert Martin - Moscow: Peter, 2018 .-- 352 p.
4. Software architecture in practice: a tutorial / L. Bass, P. Clements, R. Katzma - 2nd ed. - Moscow: Peter, 2006 .-- 576 s
5. Wikipedia.org Consolidated Encyclopedia Hub [Electronic resource] Access mode: URL https://ru.wikipedia.org/wiki/%D0%A5%D0%B0%D0%B1 (date of access 11.04.2021).
6. Kruchinin, V. V. Development of network applications: a tutorial / V. V. Kruchinin. - Moscow: TUSUR, 2013 .-- 120 p. - Text: electronic // Lan: electronic library system. - URL: https://e.lanbook.com/book/110372 (date of access: 11.04.2021). - Access mode: for authorization users.