|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Приднестровский государственный университет им. Т.Г. Шевченко | | | | | | | | | |
| Инженерно-технический институт | | | | | | | | | |
| Инженерно-технический факультет | | | | | | | | | |
| Кафедра программного обеспечения вычислительной техники  и автоматизированных систем | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | | | |
|  |  |  |  |  |  |  | | | |
|  |  |  |  |  |  |  | | | |
|  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  | | | |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | |
|  | | | | | | | | | |
| **НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА** | | | | | | | | | |
|  | | | | | | | | | |
| (1 семестр) | | | | | | | | | |
|  | | | | | | | | | |
|  | | | | | | | | | |
|  | | | | | | | | | |
|  | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  | Студент | |  |  |  |  | Михайлов | | |
|  | группы ИТ22ВР68ПИ | | |  |  |  | Сергей Юрьевич | | |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  | Научный руководитель, | | |  |  |  |  | | |
|  | доцент Кирсанова А.В. | | |  |  |  |  | | |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| Тирасполь, 2022 | | | | | | | | | |

**СЕРВЕР ОРИЕНТИРОВАНАЯ РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ НА ПРИМЕРЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ БАНКА**

Михайлов С.Ю., магистрант, доцент Кирсанова А.В.

***Аннотация.*** *В статье рассматривается сервер-ориентированный подход к разработке мобильных приложений. Рассмотрены основные преимущества и недостатки сервер-ориентированного подхода к разработке мобильных приложений. Обоснован выбор фреймворка flutter для разработки клиентской части, а также Asp.net core и микро-сервисной архитектуры для разработки серверной части. Kubernetes как платформы оркестрации.*

Сервер ориентированная разработка – архитектурный стиль, который используется в мобильных и веб-приложениях. Он основан на том, что сервер определяет как должен выглядеть интерфейс пользователя и какие данные должны быть отображены на экране.

В отличии от традиционного процесса разработки который делает пользовательский интерфейс встроенным в приложение, что приводит к негибкости и сложности обновления*.* Данный подход оптимален для мобильных приложений, которым часто требуется A/B-тестирование, добавление маркетинговых баннеров или ребрендинг, поэтому имеет смысл уделить внимание адаптивной разработке пользовательского интерфейса.

При использовании данного подхода сервер отвечает за управление интерфейсом пользователя и обмен данными с клиентской частью приложения. Сервер может отправлять новые визуальные элементы клиенту для обновления интерфейса приложения на основе динамически изменяющихся данных.

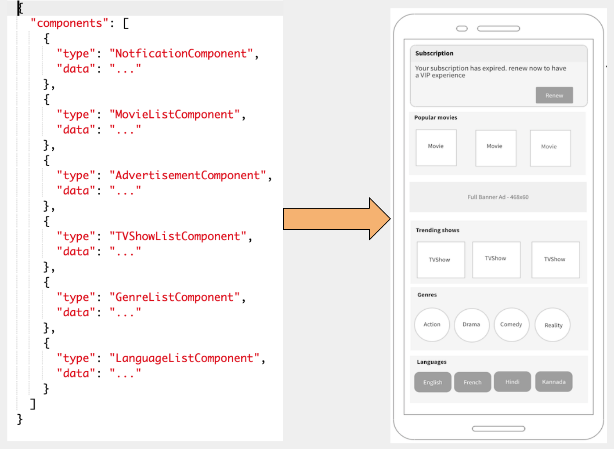


Рисунок. Сервер ориентированная разработка

Таким образом основным преимуществом является возможность изменения интерфейса мобильного приложения без необходимости его повторного развёртывания. Это может быть полезно для приложений, которым необходимо отображать динамический и часто обновляемый контент.

Существует и ряд недостатков, которые следует учитывать при использовании:

1. Зависимость от сетевого соединения. Большее количество сетевых запросов для получения и отображения данных, что может снизить эффективность приложения при работе с большими объемами данных.
2. Тестирование приложения может быть сложнее, так как необходимо учитывать работу сервера и сетевое соединение. Необходимо тестировать отправку запросов на сервер, обработку данных на сервере, а также отображение данных на устройстве. Это может требовать наличия модульных тестов для каждого компонента приложения и интеграционных тестов для проверки работы всего приложения в целом. Кроме того, может потребоваться тестирование различных сценариев сетевых условий, таких как низкая скорость соединения или отсутствие соединения.

В качестве фреймворка для разработки мобильного приложения выбран *Flutter*. *Flutter* это комплект средств разработки и фреймворк с открытым исходным кодом для создания мобильных приложений под *Android* и *iOS*, веб-приложений, а также настольных приложений под *Windows*, *macOS* и *Linux* с использованием языка программирования *Dart*, разработанный и развиваемый корпорацией *Google.*

Преимуществами *Flutter* являются:

1. Быстрая разработка: *Flutter* использует компилятор *Dart*, который позволяет создавать приложения с быстрой скоростью разработки. Кроме того, *Flutter* имеет широкий набор готовых виджетов, которые можно использовать для создания интерфейса приложения.
2. Кросс-платформенность: *Flutter* позволяет создавать приложения для *iOS*, *Android*, *Windows*, *Linux* что упрощает разработку и позволяет развивать одно приложение для двух основных мобильных платформ.
3. Высокая производительность: Код *Flutter* компилируется в машинный код ARM или *Intel*, а также в *JavaScript* для быстрой работы на любом устройстве.[1]

Основным недостатком *Flutter* являетсянебольшое сообщество разработчиков т.к. это относительно новый фреймворк, что может усложнить решение проблем.

Для разработки серверной части приложения выбран *ASP.NET Core* и микро-сервисная архитектура.[2] *ASP.NET Core* является кроссплатформенной, высокопроизводительной средой с открытым исходным кодом для создания современных приложений и имеет ряд преимуществ, которые делают его популярным выбором для разработки веб-приложений и сервисов:

1. Кросс-платформенность: *ASP.NET Core* может работать на *Windows*, *Linux* и *MacOS*, что дает возможность разрабатывать и развертывать веб-приложения на различных платформах.
2. Открытый исходный код: *ASP.NET Core* является открытым исходным кодом, что дает возможность разработчикам изучать и изменять исходный код фреймворка.

Микро-сервисная архитектура – это подход к разработке программного обеспечения, который предполагает разбиение системы на небольшие, независимые компоненты, которые называются микро-сервисами. Каждый микро-сервис специализируется на выполнении определенной функции и разрабатывается и поддерживается независимо от других микро-сервисов. [3]

Микро-сервисная архитектура имеет ряд преимуществ, таких как легкость масштабирования, развертывания и разработки, а также улучшенную гибкость и надежность системы. Каждый микро-сервис может использовать тот язык программирования и те технологии, что удобны команде, которая его разрабатывает.

Однако, использование микро-сервисной архитектуры также сопряжено с рядом вызовов, таких как сложность управления и отладки, сложность реализации распределённых транзакций и пр.

В качестве платформы развёртывания серверной части приложения выбран *Kubernetes*. *Kubernetes* это портативная расширяемая платформа с открытым исходным кодом для управления контейнеризованными рабочими нагрузками и сервисами, которая облегчает как декларативную настройку, так и автоматизацию. У платформы есть большая, быстро растущая экосистема[4].

Сервисы, поддержка и инструменты *Kubernetes* широко доступны. *Kubernetes* использует контейнерную технологию, такую как *Docker*, для упаковки приложений вместе с всеми необходимыми зависимостями и ресурсами, чтобы они могли работать на любой машине. Это обеспечивает простоту и переносимость приложений, а также упрощает их управление и масштабирование.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Flutter Developmet [Электронный ресурс] // Режим доступа: https://flutter.dev//

2. Overview of ASP.NET Core [Электронный ресурс] // Режим доступа: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-7.0

3. Microservices [Электронный ресурс] // Режим доступа: https://microservices.io/

4. Kubernetes [Электронный ресурс] // Режим доступа: https://kubernetes.io

5. .NET Microservices: Architecture for Containerized .NET Applications [Электронный ресурс] // Режим доступа: <https://learn.microsoft.com/en-us/dotnet/architecture/microservices/>

6. .NET Application Architecture [Электронный ресурс] // Режим доступа: https://github.com/dotnet-architecture

**Summary**

The article discusses the server-driven ui approach to mobile development on the example of a bank's mobile application and the application architecture. In particular, such tools and approaches as microservice architecture, features of flutter frameworks, asp.net core and kubernetes containerization platforms.

SERVER-ORIENTED DEVELOPMENT OF A MOBILE APPLICATION ON THE EXAMPLE OF A BANK MOBILE APP

Mihailov S.Y., Kirsanova A.V.