|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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, | | |  |  |  |  | | |
|  | см. приказ | | |  |  |  |  | | |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| Тирасполь, 2022 | | | | | | | | | |

**//TODO МОБИЛЬНОЕ ПРИЛОЖЕНИЕ ОСНОВАНОЕ НА ПРИНЦИПАХ *SERVER-DRIVER UI***

Михайлов С.Ю., магистрант, //todo

***Аннотация.*** *В статье рассматриваются //todo* Аннотация

*Server-driven UI (User Interface)* 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*. Server-driven UI* 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 подхода *Server-driven UI* 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. Таким образом основным преимуществом подхода *Server-driven UI* 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 *Server-driven UI*:

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 с *Server-driven UI* 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. //todo ссылка на wiki, flutter.dev

Существует и ряд недостатков, которые следует учитывать при использовании *Flutter*.

1. Небольшое сообщество разработчиков: *Flutter* относительно новый фреймворк, и поэтому он имеет небольшое сообщество разработчиков, что может сложнее найти решения проблем или получить помощь в разработке.
2. Некоторые устройства могут не поддерживаться: *Flutter* может не работать на некоторых устройствах, которые устарели или имеют старую версию операционной системы.

Для разработки серверной части приложения выбран ASP.NET Core и язык программирования с#, микросервисная архитектура. ASP.NET Core является кроссплатформенной, высокопроизводительной средой с открытым исходным кодом для создания современных приложений. (А c# просто классный.//todo rm).

ASP.NET Core 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.

Микросервисная архитектура – это подход к разработке программного обеспечения, который предполагает разбиение системы на небольшие, независимые компоненты, которые называются микросервисами. Каждый микросервис специализируется на выполнении определенной функции и разрабатывается и поддерживается независимо от других микросервисов.

Микросервисная архитектура имеет ряд преимуществ, таких как легкость масштабирования, развертывания и разработки, а также улучшенную гибкость и надежность системы. Однако, использование микросервисной архитектуры также сопряжено с рядом вызовов, таких как сложность управления и отладки, сложность реализации распределённых транзакций и пр.

//заключение

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. https://learn.microsoft.com. [Электронный ресурс] // Режим доступа: https://learn.microsoft.com/ru-ru/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-7.0

2. https://flutter.dev/ [Электронный ресурс] // Режим доступа: https://flutter.dev/

**Summary**