ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

***«*САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»**

|  |
| --- |
| Институт компьютерных наук и технологий  Высшая школа программной инженерии |
|  |

**ИНДИВИДУАЛЬНЫЙ ПЛАН (ЗАДАНИЕ И ГРАФИК)**

**ПРОВЕДЕНИЯ ПРАКТИКИ**

|  |
| --- |
| Коновалов Александр Анатольевич |

|  |
| --- |
| Направление подготовки (код/наименование) 09.03.04 «Программная инженерия» |
| Профиль (код/наименование) 09.03.04\_01 «Технология разработки и сопровождения качественного программного продукта» |
| Вид практики: научно-исследовательская работа |
| Тип практики: распределенная |
| Место прохождения практики ФГАОУ ВО «СПбПУ», ИКНТ, ВШПИ, СПб, ул.Политехническая, 29 |

|  |
| --- |
| Руководитель практической подготовки от ФГАОУ ВО «СПбПУ»: |

|  |  |
| --- | --- |
| Котлярова Лина Павловна, старший преподаватель ВШПИ ИКНТ | Петров Александр Владимирович, старший преподаватель ВШПИ ИКНТ |

*(Ф.И.О., уч.степень, должность)*

|  |
| --- |
| Руководитель практической подготовки от профильной организации: - |

**Рабочий график проведения практики**

Сроки практики: с02.02.23 по 14.04.23



|  |  |  |  |
| --- | --- | --- | --- |
| **№ п/п** | **Этапы (периоды) практики** | **Вид работ** | **Сроки прохождения этапа (периода) практики** |
| 1 | Организационный этап | Установочная лекция для разъяснения целей, задач, содержания и порядка прохождения практики, инструктаж по технике безопасности, выдача сопроводительных документов по практике | 02.02 |
| 2 | Основной  этап | Разработка клиентской части веб-приложения для настройки функционала IPTV-приставок провайдером | 03.02-13.04 |
| 3 | Заключительный этап | Защита отчета по практике | 14.04 |

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Коновалов А. А. /

Руководитель практической подготовки

от ФГАОУ ВО «СПбПУ» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Котлярова Л. П. /

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

***«*САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»**

|  |
| --- |
| Институт компьютерных наук и технологий  Высшая школа программной инженерии |
|  |

**Отчет о прохождении научно-исследовательской работы**

|  |
| --- |
| Коновалов Александр Анатольевич |

|  |
| --- |
| 4 курс, 3530904/90105 |

|  |
| --- |
| 09.03.04 «Программная инженерия» |

|  |
| --- |
| **Место прохождения практики:** ФГАОУ ВО «СПбПУ», ИКНТ, ВШПИ, |

|  |
| --- |
| СПб, ул.Политехническая, 29 |

|  |
| --- |
| **Сроки практики:** 02.02.23-14.04.23 |

|  |
| --- |
| **Руководитель практической подготовки от ФГАОУ ВО «СПбПУ»:** |

|  |  |
| --- | --- |
| Котлярова Лина Павловна, старший преподаватель ВШПИ ИКНТ | Петров Александр Владимирович, старший преподаватель ВШПИ ИКНТ |

|  |
| --- |
| **Руководитель практической подготовки от профильной организации:** - |

|  |
| --- |
| *Колычев Александр Владимирович* |

|  |
| --- |
| **Оценка: зачтено/не зачетно** |

|  |
| --- |
| Руководитель практической подготовки  от ФГАОУ ВО «СПбПУ»: / Котлярова Л.П. / |

|  |
| --- |
| Руководитель практической подготовки  от профильной организации: / Колычев А. В. / |

|  |
| --- |
| Обучающийся: / Коновалов А. А. / |

|  |
| --- |
| Дата: 14.04.23 |

# Оглавление

[Оглавление 2](#_Toc132301446)

[Список обозначений и сокращений 3](#_Toc132301447)

[1 Постановка задачи 4](#_Toc132301448)

[2 Обоснование актуальности работы 5](#_Toc132301449)

[3 Обзор существующих решений 6](#_Toc132301450)

[4 Обоснование выбора технологий и средств разработки 9](#_Toc132301451)

[5 Описание планируемой архитектуры 11](#_Toc132301452)

[6 Список используемых источников информации 14](#_Toc132301453)

# Список обозначений и сокращений

**Dart** – язык программирования созданный компанией Google [1]. Dart является статически типизированным объектно-ориентированным языком программирования [2].

**Flutter** – фреймворк с открытым исходным кодом, созданный Google. Он используется для разработки кроссплатформенных приложений для Android, iOS, Linux, macOS, Windows, Google Fuchsia и Web-браузеров из единой кодовой базы с помощью языка программирования Dart [3].

**IP** – маршрутизируемый протокол сетевого уровня стека TCP/IP.

**IPTV** – Internet Protocol Television (рус. Интерактивное телевидение). Технология цифрового телевидения в сетях передачи данных по протоколу IP [4].

**IPTV**-**Middleware** – промежуточное серверное ПО, позволяющее управлять всеми компонентами системы IPTV [5].

**Json** – текстовый формат обмена данными, основанный на синтаксисте JavaScript [6].

**Provisioning** – метод автоматической централизованной настройки IPTV-приставок [7].

**UI** – User Interface (рус. Пользовательский интерфейс)

**UX** – User Experience (рус. Пользовательский опыт)

**XML** – Расширяемый язык разметки [8]

**ОС** – Операционная система

**ПО** – Программное обеспечение

# Постановка задачи

Целью научно-исследовательской работы является создание клиентской части веб-приложения для настройки IPTV-приставок провайдером с помощью технологии Provisioning, разработанной компанией TVIP[9]. Веб-приложение должно заменить существующий способ настройки, осуществляемый составлением XML файла с настройками, и предоставить удобный интерфейс для администраторов операторов связи.

Для достижения поставленной цели планируется разработать веб-приложение с помощью кроссплатформенного фреймворка Flutter от компании Google, созданного на основе языка Dart.

# Обоснование актуальности работы

В настоящее время существует множество способов доставки телевизионного контента от оператора цифрового телевидения до конечного пользователя, зрителя. Одним из решений является телевидение по протоколу IP (Интерактивное телевидение) (англ. Internet Protocol Television) (IPTV). В связи с развитием интернета эта технология становится все более популярной, так как обладает многими преимуществами, например воспроизведением телепрограмм на любом устройстве, удовлетворяющим системным требованиям и подключенном к сети интернет, доставка пользователю дополнительного контента и др.

Ключевым компонентом технологии IPTV является IPTV-Middleware[5], именно благодаря ему у операторов связи есть возможность управлять контентом, предоставляемым пользователям. Однако, помимо контента, операторы цифрового телевидения заинтересованы в контроле за устройствами клиентов.

В контроль над устройством входит непосредственно само управление приставками. Примером которого может служить блокировка функционала приставки в случае расторжения договора об оказании услуг, это актуально, так как основная часть приставок предоставляется оператором связи вместе с услугами телевещания. В таком случае вместе с утратой основного контента оборудование пользователя лишается всех остальных возможностей.

Автоматизация настройки устройств является важной частью контроля, она облегчает работу как пользователю, так и оператору связи. Автоматизация избавляет специалистов от ручной настройки, так как позволяет получать все необходимые настройки автоматически по сети, чем ускоряет и упрощает подключение новых абонентов. Со стороны пользователя пропадает необходимость постоянно следить за обновлениями ПО или настраивать отдельные компоненты системы, что улучшает пользовательский опыт.

В дополнение, такой контроль позволяет операторам связи, приобретающим данное ПО, собирать статистику с устройств пользователей. А также задавать брендирование – настраивать пользовательский интерфейс в соответствии с общим дизайн-кодом компании оператора.

Все описанные задачи позволяет решить метод автоматической централизованной настройки функционала IPTV-приставок – Provisioning[7]. На данном этапе вся настройка производится посредствам конфигурирования XML файла с настройками, однако для такого широкого спектра задач необходим инструмент с удобным пользовательским интерфейсом, который позволит легко производить настройку администраторам операторов связи. В качестве такого инструмента выступает веб-приложение, созданию клиентской части которого и посвящена выпускная квалификационная работа.

# Обзор существующих решений

Описываемый метод централизованной настройки устройств впервые был разработан и представлен компанией TVIP[9], однако позже появились его аналоги от других компаний, которые предоставляют свои клиентские приложения, облегчающие работу с их инструментом.

Так, компания ELTEX предоставляет решение «Сервер конфигурации - ACS (Automatic Configuration Server)» [10] – веб-приложение, позволяющее сетевым администраторам удаленно управлять и настраивать сетевые устройства ELTEX. Это приложение упрощает процесс конфигурирования и экономит время настройки. Одной из ключевых особенностей ELTEX ACS является возможность автоконфигурации. Эта функция позволяет сетевым администраторам применять заранее заданные шаблоны конфигурации сразу к нескольким устройствам, что экономит время и снижает вероятность ошибок.

Еще одно решение – Mvision[11], предлагаемое компанией Microimpuls. Это веб-приложение, предназначенное для автоматизации и упрощения процесса управления и настройки абонентского оборудования. MVision предоставляет централизованное управление настройками абонентских устройств, с использованием удобного веб-интерфейса. С его помощью можно проводить настройку различных параметров, таких как каналы, плейлисты, настройки аудио и видео, управление доступом и другие. MVision обеспечивает удобный и эффективный процесс управления абонентским оборудованием, что способствует оптимизации работы провайдеров IPTV-сервисов.

Исходя из области применения такого рода веб-приложений основными критериями для оценки решений будем считать:

1. *Доступные для управления и настройки устройства*

Определяет какими устройствами можно управлять (IPTV-приставки, Маршрутизаторы, и др.)

1. *Поддерживаемые платформы*

На каких платформах, устройствах, браузерах возможно запустить приложение.

1. *Наличие адаптивного дизайна*

Определяет могут ли UI-элементы приложения быть оптимально отображены на разных устройствах с различными размерами экранов, разрешениями и ориентациями (например, на компьютерах, планшетах и мобильных устройствах).

1. *Потенциал создания мобильного приложения*

Определяет возможность и трудоемкость создания мобильного приложения, дублирующего функционал веб-приложения. При расчете трудоемкости учитывается возможность переиспользования уже имеющейся бизнес-логики и UI-компонентов

1. *Наличие локализации*
2. *Наличие средств сбора статистики*

В каком виде и по каким критериям можно просматривать статистику, возможен ли ее экспорт и т. д.

1. *Функционал*

Описывает, какой функционал инструмента для автоматизации настройки доступен в приложении.

1. *Наличие интерфейса для главного администратора*

Описывает, существует ли специальный интерфейс для главного администратора, с возможностью создания и изменения новых ролей и учетных записей администраторов.

В таблице 1 приведены результаты сравнения вышеперечисленных аналогов по описанным критериям. Таблица будет дополнена данными при написании выпускной квалификационной работы.

Таблица 1 Сравнение веб-приложений для методов централизованной настройки IPTV-приставок

|  |  |  |
| --- | --- | --- |
| Критерий | ACS | MVision |
| Доступные для управления и настройки устройства | Пользовательские устройства Eltex (IPTV-приставки, маршрутизаторы, коммутаторы) и совместимые устройства сторонних производителей | IPTV-приставки Microimpuls |
| Поддерживаемые платформы | Все платформы с наличием интернет-браузера | Все платформы с наличием интернет-браузера |
| Адаптивный дизайн | Нет | Нет |

|  |  |  |
| --- | --- | --- |
| Потенциал создания мобильного приложения | Необходимость создания с нуля. Отсутствие возможности переиспользования бизнес-логики и UI-компонентов | |
| Наличие локализации | Да | Да |
| Наличие средств сбора статистики | Да | Да |
| Наличие интерфейса для главного администратора | Да | Нет |
| ***Функционал*** | | |
| Возможность смены ОС приставки | Нет | Нет |
| Возможность изменения пользовательского интерфейса приставки | Нет | Нет |
| Обновление ПО приставки | Да | Да |

Исходя из сравнения аналогов и учитывая их недостатки, можно выделить необходимые требования, которые необходимо реализовать в предлагаемом решении для инструмента Provisioning[7].

# Обоснование выбора технологий и средств разработки

Для разработки веб-приложения выбран кроссплатформенный фреймворк Flutter[3], основанный на языке Dart[2]. Он позволяет создавать приложения под различные платформы, в числе которых: Web (браузеры: Chrome, Firefox, Safari, Edge), Android, IOS, GNU/Linux, Windows, MacOS.

С помощью Flutter в компании уже реализовано несколько проектов, имеются собственные закрытые библиотеки и наборы средств разработки. Также в будущем будет возможность распространить созданное решение на другие платформы, сохранив слой бизнес-логики приложения и многие визуальные компоненты. По этим причинам в качестве основного инструмента разработки выбран именно Flutter.

Средой разработки выбрана AndroidStudio[12] от компании Google. Это одна из наиболее популярных сред разработки для Flutter, она удобна в использовании и позволяет запускать и тестировать приложения на всех поддерживаемых платформах. Также в AndroidStudio встроены инструменты для оценки производительности приложения. Однако для качественной оценки скорости работы, контроля за расходом памяти, расположением элементов на экране и т. д. используется набор инструментов Flutter DevTools[13].

Для контроля версиями проекта и организации работы используется платформа GitLab[20], который работает с системой контроля версий Git[21].

Для реализации многих функций в приложение также встроены пакеты от сторонних разработчиков.

Так, для локализации веб-приложения используется сторонний пакет intl[14], он удобен в использовании и поддерживает все языки.

Контроль за состоянием приложения (state-management) осуществляется с помощью стороннего пакета Flutter BLoC[22], а для внедрения моделей – пакет Provider [15]. Также грамотный контроль за состоянием приложения, корректное обновление интерфейса и логики требуют неизменяемых моделей (классов), этого позволяет добиться пакет Freezed[16].

Пакет JsonSerializable[17], также включенный в приложение, предоставляет удобный инструмент, который автоматически генерирует Json сериализаторы и десериализаторы для классов моделей.

Для безопасного сохранения пользовательских данных между сеансами использования веб-приложения используется пакет SharedPreferences[18]. Пакет предоставляет постоянное хранилище типа ключ-значение и удобен для подключения и взаимодействия.

Тестирование приложения осуществляется с помощью встроенной библиотеки Flutter Test[19]. Она позволяет проводить модульное тестирование, тестирование пользовательского интерфейса (виджет-тесты), а также интеграционное тестирование.

# Описание планируемой архитектуры

Язык Dart является статически типизированным языком программирования, использующим принцип объектно-ориентированного программирования (ООП).

Архитектурно разрабатываемое веб-приложение представляет собой 3 слоя: UI-слой (UI-layer), слой бизнес-логики (Business-logic layer) приложения и слой работы с данными (Data-layer). Такое разбиение по слоям позволяет разграничить различные части приложения и уменьшает связность компонентов, позволяя более гибко изменять их. Зависимости слоев направленны строго внутрь, UI-слой зависит от слоя бизнес-логики, который в свою очередь зависит от слоя данных.

UI-слой отвечает за ту часть приложения, которую пользователь видит на экране, все внешние компоненты (текст, кнопки и др.), этот слой получает данные из слоя бизнес-логики. Он осведомлен о формате необходимых данных , но ничего не знает об устройстве слоя бизнес-логики и общается с ним по определенному интерфейсу (контракту). При этом слой бизнес-логики, как и другие, можно подменить в любой момент, допустим тестовыми данными, главное, чтобы он удовлетворял описанному контракту. На этом слое используются стандартные инструменты фреймворка Flutter для создания UI элементов.

В предлагаемом решении на этом слое содержатся все виджеты (составляющие UI во Flutter), а также экраны.

Слой бизнес-логики отвечает за прием и обработку данных от слоя данных, суть обмена данными такая же, как UI-слой общается со слоем бизнес-логики. Также этот слой ответственен за навигацию внутри приложения. Для этого используются инструменты Flutter, сторонние библиотеки, такие как Flutter BLoC[22], и закрытые библиотеки, разработанные силами компании, облегчающие контроль за состоянием приложения.

В приложении данный слой представлен в виде классов Bloc, наследующих Flutter BLoC. Они прослушивают изменения состояния на слое данных, реагируют на это, изменяют собственное состояние и предоставляют его UI слою. На этом слое особенное место занимают глобальные Bloc’и, отвечающие за статус аутентификации пользователя (AuthBloc) и навигацию (NavigatorBloc), так как относятся непосредственно ко всему приложению, а не к отдельному экрану или модулю.

Самым нижним слоем выступает слой данных. На этом слое сосредоточена работа по получению данных с сервера, обработке возникающих ошибок и представлении полученных данных в виде объектов, которые передадутся ниже, на слой бизнес-логики.

В приложении на этом слое находятся 3 типа объектов. Repository – класс, описывающий доступные запросы к серверу; HttpClient – класс, реализующий логику обмена данными с сервером, посредствам http-запросов; ApiBloc – специальные классы, основанные на библиотеке Flutter BLoC, которые ответственны за запрос и получение данных из Repository и HttpClient, их последующей обработке, и предоставления нового состояния (корректного состояния, состояния ожидания выполнения запроса, состояния ошибки и др.), на которое будет опираться слой бизнес-логики.

На рисунке 1 представлена схема планируемой архитектуры веб-приложения. Линиями разделены описанные выше слои, стрелками показаны зависимости между слоями и их элементами. Также отдельными блоками выделен удаленный сервер (Remote Server), к которому приложение обращается посредствам запросов (Request), и принимает от него ответы (Response) и локальное постоянное хранилище Shared Preferences.

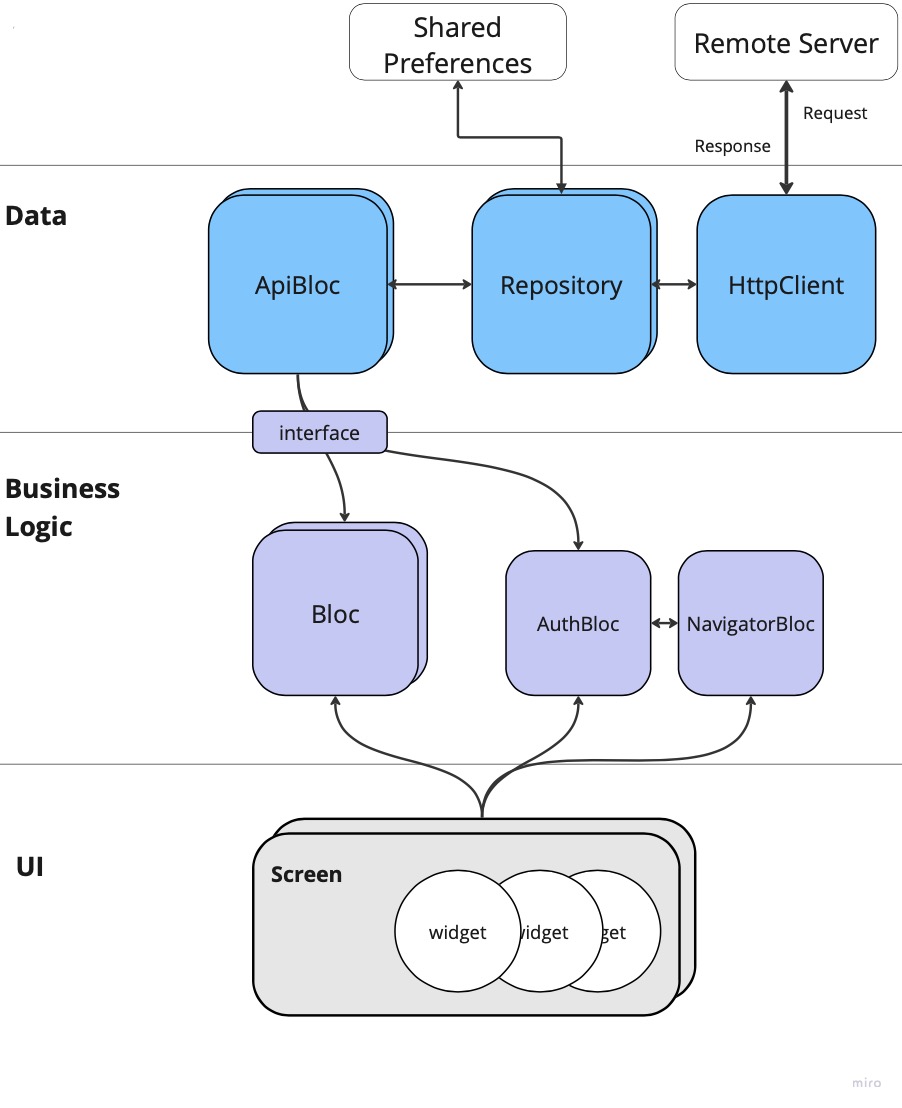


Рис. 1 Схема архитектуры веб-приложения

# Список используемых источников информации

1. About Google [Электронный ресурс]. URL: https://about.google/ (Дата обращения: 10.12.2022).
2. Dart Language [Электронный ресурс]. URL: https://dart.dev/ (Дата обращения: 10.12.2022).
3. Flutter [Электронный ресурс]. URL: https://flutter.dev/ (Дата обращения: 10.12.2022).
4. IPTV [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/IPTV (Дата обращения: 10.12.2022).
5. Что из себя представляет IPTV Middleware? [Электронный ресурс]. URL: https://habr.com/ru/post/168891/ (Дата обращения: 10.12.2022).
6. Json [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/JSON (Дата обращения: 10.12.2022).
7. TVIP Provisioning [Электронный ресурс]. URL: https://wiki.tvip.tv/en/provisioning (Дата обращения: 10.12.2022).
8. XML [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/XML (Дата обращения: 10.12.2022).
9. Компания TVIP [Электронный ресурс]. URL: https://www.tvip.tv/?ysclid=ld7l47v8h3574784742 (Дата обращения: 10.12.2022).
10. ELTEX.ACS [Электронный ресурс]. URL: https://eltex-co.ru/catalog/management/eltex-acs/ (Дата обращения: 10.12.2022).
11. Microimpuls MVision [Электронный ресурс]. URL: https://microimpuls.com/mvision.html?ysclid=lf6m00o1b7524345843 (Дата обращения: 10.12.2022).
12. Android Studio [Электронный ресурс]. URL: https://developer.android.com/studio (Дата обращения: 10.12.2022).
13. DevTools [Электронный ресурс]. URL: https://docs.flutter.dev/development/tools/devtools/overview (Дата обращения: 10.12.2022).
14. Internationalizing Flutter apps [Электронный ресурс]. URL: https://docs.flutter.dev/development/accessibility-and-localization/internationalization (Дата обращения: 10.12.2022).
15. Provider [Электронный ресурс]. URL: https://pub.dev/packages/provider (Дата обращения: 10.12.2022).
16. Freezed [Электронный ресурс]. URL: https://pub.dev/packages/freezed (Дата обращения: 10.12.2022).
17. JSON and serialization [Электронный ресурс]. URL: https://docs.flutter.dev/development/data-and-backend/json (Дата обращения: 10.12.2022).
18. Shared Preferences [Электронный ресурс]. URL: https://pub.dev/packages/shared\_preferences (Дата обращения: 10.12.2022).
19. Testing Flutter apps [Электронный ресурс]. URL: https://docs.flutter.dev/testing (Дата обращения: 10.12.2022).
20. GitLab [Электронный ресурс]. URL: https://about.gitlab.com/ (Дата обращения: 10.12.2022).
21. Git [Электронный ресурс]. URL: https://git-scm.com/ (Дата обращения: 10.12.2022).
22. Flutter BLoC [Электронный ресурс]. URL: https://pub.dev/documentation/flutter\_bloc/latest/ (Дата обращения: 10.12.2022).