Анисимов Валерий Алексеевич

Тема: Мобильное приложение структурного подразделения МАИ.

РЕФЕРАТ

Выпускная квалификационная работа бакалавра состоит из 39 страниц, 18 рисунков, 0 таблиц, 16 использованных источников, 1 приложений.

ПРОГРАММИРОВАНИЕ, ФРОНТЕНД, FLUTTER, DART, МОБИЛЬНАЯ РАЗРАБОТКА, ANDROID, IOS, CROSS-PLATFORM, FRAMEWORK, FRONT-END.

В первом разделе рассматриваются: постановка задачи, формулировка проблемы. Описаны: предполагаемый способ решения поставленной задачи, особенности фреймворка flutter и языка программирования dart.

Во втором разделе приведены теоретические предпосылки к решению задачи и особенности практической реализации решения. Описан стек используемых технологий и его преимущества.

В третьем разделе выпускной квалификационной работы рассмотрены результаты, представленные в ходе реализации предложенного решения, проведён анализ и оценка эффективности практической реализации, выработаны рекомендации по использованию и развитию разработанного в ходе работы подхода.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

Фреймворк - (с англ. framework — «каркас, структура») — заготовка, готовая модель для быстрой разработки.

Flutter – 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.

Dart – представляет язык программирования общего назначения, который предназначен прежде всего для разработки веб-приложений (как на стороне клиента, так и на стороне сервера) и мобильных приложений.

Android – Android — операционная система для смартфонов, планшетов, электронных книг, цифровых проигрывателей и других устройств.

iOS – мобильная операционная система для смартфонов, электронных планшетов, носимых проигрывателей, разрабатываемая и выпускаемая американской компанией Apple.

JIT (с англ. just-in-time) компилятор – реализует метод компиляции программы прямо во время её работы.

AOT (с англ. ahead-of-time) компилятор – использует метод компиляции перед исполнением.

Elibrary - российская научная электронная библиотека, интегрированная с Российским индексом научного цитирования.

Виджет (с англ. Widget) - центральным элементом приложения на Flutter. Фактически это визуальные компоненты, из которых состоит графический интерфейс.

Интерфейс - способ и средства взаимодействия пользователя с программами.

Микросервисная архитектура - подход, при котором единое приложение строится как набор небольших сервисов.

API (Application Programming Interface) – это совокупность инструментов и функций для обеспечения межпрограммного взаимодействия.

ВВЕДЕНИЕ

Деятельность научной лаборатории включает в себя множество типовых процессов, организация которых требует значительных временных и материальных затрат. Рассмотрим деятельность лаборатории, создаваемой на базе института №8 Московского Авиационного Института.

Функционирование вышеуказанной лаборатории содержит в себе такие процессы как: публикационная деятельность сотрудников лаборатории; администрирование и распределение ресурсов лабораторного оборудования; организация рабочей деятельности сотрудников; проектная деятельность; популяризация деятельности лаборатории для привлечения внимания широкого круга людей к научной активности [4, 15]. При этом, каждый из этих процессов в силу своей неоптимизированности отнимает значительное количество времени у сотрудников лаборатории, снижая их производительность и не давая сосредоточиться на научной деятельности [13]. К примеру, полностью отсутствует механизм планирования и распределения вычислительных мощностей лабораторного оборудования, затягивая процесс производства научной деятельности на этапе получение доступа к указанным мощностям [13, 15].

Оптимизация приведённых типовых процессов позволит сократить время их выполнения и сфокусировать активность сотрудников на научной деятельности [4, 13].

Целью выпускной квалификационной работы является сокращение временных затрат сотрудника научной лаборатории на выполнение типовых процессов.

В список задач входит:

- определение типовых процессов научной лаборатории,

- определение направлений совершенствования вышеуказанных процессов,

- разработка архитектуры программного решения,

- реализация приложения научной лаборатории для мобильных устройств,

- реализация микросервисов, необходимых в вышеуказанном приложении,

- оценка эффективности реализованного решения путём сравнения временных затрат на выполнение типовых процессов.

Таким образом объектом исследования является оптимизация деятельности сотрудника научного подразделения, предметом исследования – сокращение временных затрат на выполнение типовых процессов сотрудника научной лаборатории посредством их оптимизации при помощи приложения для мобильных устройств.

В результате выполнения работы автором была осуществлена реализация вышеуказанного мобильного приложения, построенного на микросервисной архитектуре [7], каждый из разделов которого соответствует типовому процессу, выполняемого в рамках научной деятельности сотрудником подразделения, и направлен на снижение временных затрат на осуществление соответствующего процесса.

В ходе анализа практических результатов реализации приложения для мобильных устройств сделан вывод, что каждый из приведённых микросервисов, внося существенные изменения в ход научный деятельности, способен значительно оптимизировать временные затраты на выполнение каждого из установленных в ходе работы типовых процессов [4, 5, 13, 15].

1. Постановка задачи

1.1 Формулировка проблемы

Для чёткого определения проблемы начнём с определения типовых процессов, совершаемых сотрудниками научной лаборатории. В список таких процессов входят [4]:

- работа с публикациями,

- проектная деятельность в составе научного подразделения,

- практические вычисления на лабораторном оборудовании,

- планирование научной деятельности,

- действия, направленные на популяризацию подразделения и научной деятельности в целом.

Разберём каждый из процессов подробнее.

Так, написание публикаций является сложным, многоэтапным процессом, который, кроме временных затрат на сам процесс написания требует отчётной деятельности и предоставления информации о текущем этапе производства руководителю или другим причастным к данной публикации сотрудникам. Кроме того, информация об уже опубликованных работах предоставлена на сторонних ресурсах со сложным, неинтуитивным интерфейсом и требует значительное количество усилий для поиска информации о конкретном авторе (например, информации о публикациях сотрудника лаборатории, необходимой руководителю подразделения) [4].

Руководителю научного подразделения также необходимо иметь быстрый доступ к информации о проектной деятельности лаборатории, текущем прогрессе по каждому из проектов, планируемых мероприятиях.

Соответственно, рядовому сотруднику требуется удобный и быстрый способ предоставлять такую информацию [4, 15].

Вычисления на лабораторном оборудовании, в силу ограниченности ресурсов, требуют быстрого и эффективного способа планирования и распределения использования вычислительных мощностей [15]. В данный момент какой-либо инструмент для этой цели отсутствует.

Эффективное и удобное планирование деятельности сотрудников научной лаборатории позволит существенно оптимизировать рабочий процесс, особенно, если инструмент для планирования позволит руководителю отслеживать и корректировать текущие задачи рядовых сотрудников [15].

Популяризация функционирования научной лаборатории также имеет немалое значение – успешная деятельность в данном направлении позволит вызвать интерес студентов и сторонних наблюдателей к научному подразделению и его проектам, в перспективе обеспечить лабораторию свежими кадрами и репутационным весом [4]. Такая деятельность в текущий момент требует значительных временных затрат на создание различного медиа контента и загрузки его на большое число сторонних платформ.

Проанализировав типовые процессы сотрудников научной лаборатории приступим к формулированию проблемы.

Работа научного подразделения для поддержания своей деятельности и эффективного им управления требует значительного организационного и отчётного обеспечения, реализация которого занимает весомый объём времени у её рядовых сотрудников, отвлекая их от непосредственного выполнения целевых задач. Те же процессы, которые не требует отчётности,

могут быть значительно оптимизированы для экономии временных ресурсов её сотрудников.

1.2 Предполагаемый способ решения

Для решения сформулированной проблемы предлагается создать единый инструмент на базе микросервисного мобильного приложения для эффективного управления и применения различных ресурсов лаборатории, включая трудовые ресурсы и время. Соответственно, для каждого типового процесса, возникающего в ходе деятельности научной лаборатории, требуется создать микросервсис, обеспечивающий существенное снижения временных затрат на выполнение связанных с ним операций, а также повышающий удобство и эффективность взаимодействия внутри подразделения [4, 5-7].

Реализация приложения предполагается на базе фреймворка flutter. Остановимся на нём более подробно, чтобы аргументировать данный выбор.

1.3 Flutter и dart

Flutter – это комплекс средств разработки и фреймворк для создания мобильных и настольных приложений, созданный на основе языка программирования dart [9]. Одной из отличительных особенностей средства разработки является кросплатформенность – flutter-приложения будут работать и на android, и на iOS-устройствах [1, 3]. Это свойство достигается компиляцией языка dart в нативный код любой из платформ, поэтому реализованные на flutter приложения работают также хорошо, как если бы они были написаны специально под одну из указанных платформ. Кроме того, flutter позволяет обращаться напрямую к открытым интерфейсам платформы: камере, файловой система и т.д. [1-3, 8]

Dart – язык программирования, позиционируемый как замена для JavaScript. Одним из преимуществ данного языка является поддержка им как just-in-time (JIT), так и ahead-of-time (AOT) компиляции [10, 11].

JIT-компиляторы реализуют метод компиляции программы прямо во время её работы, что существенно ускоряет разработку, однако, запуск программы и его работа занимают больше времени, т.к. перед непосредственно выполнениям кода компилятор анализирует и компилирует его. AOT-компиляторы, напротив, реализуют программу, которая работает и запускается быстрее и предсказуемее, однако, увеличивают время, необходимое для перезапуска программы, после внесения изменений в её код, что замедляет разработку. Соответственно, dart позволяет использовать JIT-компиляцию во время разработки программы и AOT-при её релизе. Поддержка обоих методов компиляции обеспечивает значительное преимущество для dart и flutter перед другими средствами разработки [11].

Кроме нативных языков программирования, dart можно компилировать в другие языки, например, JavaScript, что позволяет в последствии использовать код, написанный для мобильного приложения, при реализации настольного или web-приложения [10].

Таким образом, выбор данных средств разработки обеспечит довольно быструю разработку приложения, кроссплатформенность реализации и возможность портирования программы для её применения в настольном или web-интерфейсе.

1.4 Выводы

В результате анализа типовых процессов, выполняемых сотрудником научной лаборатории сформулирована проблема, заключающаяся в

неоптимизированной и неэффективной растрате временных ресурсов, требуемых на выполнение сопутствующих целевым задачам действий, таких как отчётность и организация. Было предложено решение описанной проблемы, выбрана технологическая платформа для реализации этого решения.

2 Обоснование подхода к решению задачи

2.1 Предпосылки для решения задачи

В данном разделе работы представим идею реализации каждого микросервиса, для чего проанализируем каждый соответствующий ему типовой процесс, выполняемый сотрудником научного подразделения.

Исходя из того, что процесс работы над публикациями требует как отчётности о текущем этапе деятельности и уже написанных публикациях, так и предоставления информации о среде написания статьи для других сотрудников лаборатории, участвующих в производстве работы [5, 13, 15], можно сделать вывод, что необходимо реализовать раздел приложения, содержащий в себе редактируемые списки и уже готовых публикаций, и публикаций, находящихся в стадии написания. Для предоставления информации об уже выпущенной работе, информация о каждой из них должна включать в себя следующие характеристики: название статьи, наукометрические показатели (например, число цитирований), авторы работы, издание, опубликовавшую работу, год публикации и ссылка на научную электронную библиотеку elibrary. В разделе информации о статьях, находящихся в работе, необходима следующие пункты: название работы, текущая стадия производства и ссылка на среду написания для совместной научной деятельности.

В предшествующем разделе установлено, что руководителю научного подразделения требуется быстрый доступ к информации о текущей проектной деятельности лаборатории, а сотруднику – удобный интерфейс для быстрой отчётности, касаемо этой деятельности [15]. Соответственно, раздел приложения, соответствующий данному типовому процессу должен включать в себя информацию о названиях ткущих проектов, их описание и стадию

реализации. Также необходимо предоставить список планируемых мероприятий по проектам, содержащий информацию о причастности к какому-либо проекту, описание самого мероприятия и установленную дату проведения.

Раздел, относящийся к процессу вычислений на лабораторном оборудовании, в силу отсутствия какой-либо организации последних, должен содержать в себе автоматически собираемое расписание, раскрывающее распределение вычислительных мощностей по времени между сотрудниками научного подразделения. Также раздел должен включать в себя возможность составления заявки об использовании оборудования, с последующим включением её в расписание [15].

Исходя из необходимости удобного инструмента для планирования деятельности сотрудника научного подразделения, в том числе со стороны руководителя, соответствующий раздел должен включать в себя некоторое отображение временного ряда в календарном виде, с возможностью добавления на этот временной ряд событий как рядовым сотрудником, так и его руководителем [15].

В текущий момент деятельность, по популяризации научной деятельности требует значительных временных затрат в силу большого разнообразия платформ и площадок для размещения медиа контента. Соответственно, раздел, соответствующий данному типовому процессу должен содержать инструмент для создания публикаций, включающий в себе выгрузку изображений, для последующей их отправки на какую-либо площадку для размещения, а значит, имеющий доступ к камере и внутренней памяти мобильного устройства.

Таким образом, сформулирован некоторый аналог технического задания, содержащий требования по каждому из разделов реализуемого мобильного приложения. Далее, разберём аспекты практической реализации.

2.2 Практическая реализация решения

Перед обзором практической реализации мобильного приложения следует ввести некоторые понятия, касаемо используемой среды разработки.

Реализация пользовательского интерфейса во flutter происходит при помощи виджетов – визуальных компонентов, из которых состоит графическое оформление. Виджеты описывают, как должен выглядеть тот или иной участок интерфейса, учитывая его текущую конфигурацию и положения, содержащуюся в нём информацию и другие свойства.

Виджеты могут представлять собой как простые элементы, так и сложные, состоящие из более простых. Значительная библиотека готовых виджетов предоставлена в стандартных пакетах среды разработки, однако, программист может создавать как свои вариации общедоступных, так и узкоспециализированные инструменты [8, 9].

Теперь вернёмся к реализации. Начнём с микросервиса, отвечающего за размещение информации о публикациях сотрудника.

В данном разделе необходимо иметь два подраздела, содержащих информацию о готовых работах и работах, находящийся на стадии производства. Указанные подразделы имеют довольно схожую структуру, за исключением различной информации, содержащейся в каждом из них [14].

Определим классы объектов, соответствующие готовым статьям и статьям в производстве, и их конструкторы. К примеру, класс, содержащий информацию о публикации в стадии производства приведён на рисунке 1.

```
        class PublicationInProgress {

          String name;

          String stage;

          String link;

          PublicationInProgress(this.name, this.stage, this.link);

          PublicationInProgress.withNone({

            this.name = 'null',

            this.stage = 'null',

            this.link = 'null',

          });

}
```

Рисунок 1 - Класс объекта, содержащего информацию о публикации

Также, обоим подразделам требуется виджет, отвечающий за представление элементов вышеуказанных классов в виде удобного для пользователя списка. Исключив элементы, отвечающие за эстетическое восприятие данного списка, на примере публикации в работе структура виджета описана на рисунке 2.

```
        List<Widget> _buildList(){

            return dataP.map((PublicationInProgress e) =>

                Padding(

                   child: Column(

                      children: [

                         Text(e.name,

                           style:  const TextStyle(
```

```
                        ),),
                        Text(e.stage,
                          style:  const TextStyle(
                        ),),
                        TextButton(
                           onPressed: () {/*initialUrl: e.link;*/},
                           child: Text(e.link,)
                        )
                    ],
                ),
        ).toList();
  }
```

Рисунок 2 – Структура виджета, формирующего список публикаций

Соответственно, каждый элемент представляет собой столбец информации, ссылки на различные интернет ресурсы в которых представлены при помощи текстовых кнопок, для возможности быстрого их применения [14].

При вызове из программного кода вышеописанный виджет формирует список публикаций в пользовательском интерфейсе.

Кроме отображения информации о публикациях, в подразделах необходим инструмент добавления указанной информации. Реализация такого инструмента осуществлена при помощи вызова функции showdialog при нажатии пользователем определённой кнопки. Функция showdialog возвращает класс alertdialog, содержащий текстовые поля для ввода различной информации [9]. Структура данной функции приведена на рисунке 3.

```
        showDialog(context: context, builder: (BuildContext context){
```

```
                    return AlertDialog(
                      title:  const Text('Добавить статью в работе'),
                      content:
                      SingleChildScrollView(
                        child: Column(
                          children: [
                            TextField(
                              decoration:  const InputDecoration(
                                labelText: 'Название статьи',
                              ),
                              onChanged: (String name){
                                publ.name = name;
                              },
                            ),
                            TextField(),
                            TextField(),
                          ],
                        ),
                      ),
                      actions: [
                        ElevatedButton(
                          onPressed: (){
                            setState(() {
                              dataP.add(PublicationInProgress(publ.name,          publ.stage,
publ.link));
                            });
                            Navigator.of(context).pop();
                          },
                          child:  const Text('Добавить'),
                        )
```

```
                    ],
                  );
            });
```

Рисунок 3 – Структура функции showdialog

Присутствующая на рисунке 3 переменная publ является объектом указанного на рисунке 1 класса. По мере ввода информации в текстовые поля введённые значения сохраняются в свойствах данного объекта. В итоге остаётся только сохранить новый объект в список публикаций после нажатия пользователем соответствующей кнопки.

Кроме вышеуказанного, необходимо реализовать навигацию между подразделами, для чего будет использована обычная кнопка [14]. Непосредственно реализация приведена на рисунке 4.

```
        IconButton(
                      onPressed: ()=>
                          Navigator.of(context).push(
                              MaterialPageRoute(builder: (context) => Publications())
                          ),
               icon:  const Icon(Icons.swap_horiz)),
```

Рисунок 4 – Пример реализации навигации между подразделами

Микросервис, соответствующий процессу проектной деятельности из-за схожего наполнения во много повторяет структуру раздела о научных публикациях.

Так, данные о текущих проектах и мероприятиях, как и в предыдущем микросервисе, разделены на два подраздела, для навигации между которыми используется кнопка, аналогичная кнопке, содержащейся на рисунке 4. Для

формирования каждого из подклассов используется виджет, структурно соответствующий виджету, представленному на рисунке 2. Также, схожее строение имеет функция showdialog, приведённая на рисунке 3, вызывающая интерфейс для ввода информации [2, 9, 14].

Однако, информация, хранимая внутри класса, определяющего объекты мероприятий, несколько отлична от хранимой ранее. Указанный класс представлен на рисунке 5.

```
        class Event {

          String name;

          String description;

          late DateTime dateTime;

          Event(this.name, this.description, this.dateTime);
}
```

Рисунок 5 – Класс объекта, содержащего информацию о мероприятии

Кроме привычных текстовых значений информация о предстоящем мероприятии содержит дату его проведения, представленную классом DateTime. Соответственно, ввод значения даты осуществляется не текстовым полем, как предшествующие, а при помощи виджета pickDateTime, формирующего ряд из полей для ввода даты и времени. В свою очередь pickDateTime вызывает функции showDatePicker и showTimePicker для соответствующих полей [9]. Функция pickDateTime представлена на рисунке 6.

```
        Future<DateTime?> pickDateTime(

              DateTime initialDate, {

                required bool pickDate,

              }) async {
```

```
            if (pickDate){
              final date = await showDatePicker(
                context: context,
                initialDate: initialDate,
                firstDate: DateTime(2021, 8),
                lastDate: DateTime(2100),
              );
              if (date == null) return null;
              final time = Duration(hours: initialDate.hour, minutes: initialDate.minute);
              return date.add(time);
            } else {
              final timeOfDay = await showTimePicker(
                  context: context,
                  initialTime: TimeOfDay.fromDateTime(initialDate));
              if (timeOfDay == null) return null;
              final date = DateTime(initialDate.year, initialDate.month, initialDate.day);
              final time = Duration(hours: timeOfDay.hour, minutes: timeOfDay.minute);
              return date.add(time);
            }
          }
```

Рисунок 6 – виджет pickDateTime

Функции showDatePicker и showTimePicker входят в стандартный набор инструментов среды разработки flutter и представляют собой виджеты, содержащие в себе интерфейс для ввода даты и времени соответственно [9].

Перейдём к микросервису, содержащему интерфейс для отслеживания расписания использования лабораторного оборудования и формирования соответствующих заявок.

Основным инструментом для производительных вычислений в данный момент времени являются графические видеоадаптеры (gpu), возьмём их количество равным восьми. Тогда данные о расписании оборудования будем хранить в двумерном массиве, с отображением их в интерфейсе при помощи виджета обыкновенной таблицы, одна из осей которой будет отображать временной ряд, другая – одну из используемых для вычислений gpu. Соответственно, поля в таблице будут содержать информацию о загруженности одной из gpu в определённый момент времени [14].

Обновление таблицы реализуем при помощи функции load, представленной на рисунке 7.

```
        void load() async {

            setState((){

              while (Utils.toDate(old) != Utils.toDate(DateTime.now())){

                for (var j = 0; j < 13; j++){

                  for (var i = 0; i < 8; i++){

                    gpuTable[i][j] = gpuTable[i][j+1];

                  }

                }

                for (var i = 0; i < 8; i++){

                  gpuTable[i][13] = "empty";

                }

                old = old.add(const Duration(days: 1));

              }

            });

  }
```

Рисунок 7 – Функция load

Функция представляет собой простой алгоритм пошагового смещения столбцов таблицы, пока хранимое значение даты не станет равным текущей дате.

Также, необходим инструмент для инициации заявки об использовании лабораторного оборудования. Реализацию осуществим при помощи функции showDialog, аналогично её применению на рисунке 3. Для формирования заявки пользователю необходимо будет указать количество требуемых ему для вычислений графических адаптеров и длительность их использования.

Полученные данные применяем для автозаполнения таблицы, производимой функцией, структура которой описана на рисунке 8.

```
void addToTable(int nGpu, int time){

    int freeGpuInRow;

    bool lineOk;

    int linesOk;

    int done;

    dynamic i, j;

    for(j = 0; j < 14; j++){

      freeGpuInRow = 0;

      if(j + time < 15){

        for (i = 0; i < 8; i++){

          if(gpuTable[i][j] == "empty") {freeGpuInRow++;}

        }

        if (freeGpuInRow >= nGpu){

          linesOk = 0;

          for (i = 0; i < 8; i++){

            lineOk = true;

            for (var t = 0; t < time; t++){
```

```
                    if (gpuTable[i][j + t] != "empty") {lineOk = false;}
                  }
                  if (lineOk) {linesOk++;}
                }
                if(linesOk >= nGpu){
                  done = 0;
                  for (i = 0; i < 8; i++){
                    lineOk = true;
                    for (var t = 0; t < time; t++){
                      if (gpuTable[i][j + t] != "empty") {lineOk = false;}
                    }
                    if (lineOk) {
                      for (var t = 0; t < time; t++){
                        gpuTable[i][j + t] = "user";
                      }
                      done++;
                      if (done == nGpu) {return;}
                    }
                  }
                }
              }
            }
          }
        }
```

Рисунок 8 – Функция addToTable

Функция осуществляет автозаполнение таблицы расписания, руководствуясь следующим алгоритмом:

- Проходя по значениям, соответствующим каждой дате, определяет, в какую момент времени количество свободных графических устройств удовлетворяет запросу пользователя;

- Если при этом нужное количество gpu свободны от вычислений требуемое пользователем время, то вычислительные мощности на указанное время закрепляются за пользователем;

- Иначе проверяется следующая возможная дата.

Микросервис для планирования научной деятельности реализуем при помощи стандартного пакета syncfusion_flutter_calendar среды разработки flutter. Данный пакет включает в себя набор виджетов и инструментов для отображения временного ряда календарного вида в различном масштабе [1, 9].

Для отображения событий, создаваемых пользователем, введём класс, приведённый на рисунке 10.

```
class Event {
  final String title;
  final DateTime from;
  final DateTime to;
  const Event({
    required this.title,
    required this.from,
    required this.to,
  });
}
```

Рисунок 10 – Класс Event

Данные о планируемых события будем хранить в списке объектов класса Event.

Так как объект класса Event содержит поля типа DateTime, для добавления пользователем нового события к отображаемым на временном ряду воспользуемся виджетом pickDateTame, аналогично указанному на рисунке 6 [9].

В разделе, соответствующему процессу популяризации научной деятельности, создадим для пользователя функционал, позволяющий создавать небольшие публикации для отправки их на какую-либо медиа-площадку.

Поставку медиа-контента в приложение реализуем при помощи функции pickImage, приведённой на рисунке 11 [9, 14].

```
        Future pickImage(ImageSource source) async {

            try {

              final image = await ImagePicker().pickImage(source: source);

              if (image == null) return;

              final imageTmp = File(image.path);

              setState((){

                this.image = imageTmp;

              });

            } on PlatformException catch (e) {

              print('Failed to pick image $e');

            }

  }
```

Рисунок 11 – Функция pickImage

Вышеуказанная функция реализована благодаря стандартному пакету среды разработки flutter, позволяющему получать доступ к камере или локально хранимым файлам мобильного устройства [2, 9].

Так же предоставим пользователю возможность прикрепить к медиа-файлу некоторое описание посредством текстового поля.

В результате имеем легковесную публикацию, готовую к отправке на какую-либо площадку для размещения.

2.3 Стек технологий

Рассмотрим технологии, используемые при решении поставленной задачи.

Реализация приложения, как уже указывалось выше, была осуществлена посредством среды разработки и фреймворка flutter, основывающегося на языке программирования dart. Использование данной среды, благодаря её отличительным особенностям, таким как использование just in time (JIT) компиляции в процессе написания интерфейса, позволило существенно снизить длительность разработки. Также, приложение, написанное на flutter, может быть выпущено как для Android, так и для iOS-мобильный устройств. Кроме того, не составит труда в последствии реализовать WEB или настольное приложение на базе уже написанного [3, 9, 12].

Хранение данных, используемых в приложении, подразумевается как во внутренней памяти мобильного устройства, так и на облачном хранилище посредством сервиса firebase.

Firebase представляет собой систему управления базами данных, позволяющая хранить и синхронизировать между пользователями различные

данные. Общение клиентского приложения с базой данных осуществляется посредством API (Application Programming Interface) – одним из протоколов взаимодействия [16].

Использование firebase позволяет предоставить сервису back-end часть разработки мобильного приложения и сосредоточиться на front-end составляющей.

2.4 Выводы

В данном разделе работы сформулированы теоретические предпосылки к решению задачи, на основании которых осуществлена практическая реализация предложенного решения. В ходе описания реализации приведены фрагменты программного кода и ключевых моментов каждого из микросервисов мобильного приложения. Также в разделе проанализированы преимущества применения в ходе разработки используемого стека технологии.

3. Интерпретация результатов

3.1 Представление результатов

В данном разделе выпускной квалификационной работы для предоставления результатов практической реализации мобильного приложения будет приведён графический интерфейс каждого из микросервисов.

Начнём с интерфейса микросервиса, отвечающего за процесс работы с публикациями. Интерфейс отображён на рисунке 12.

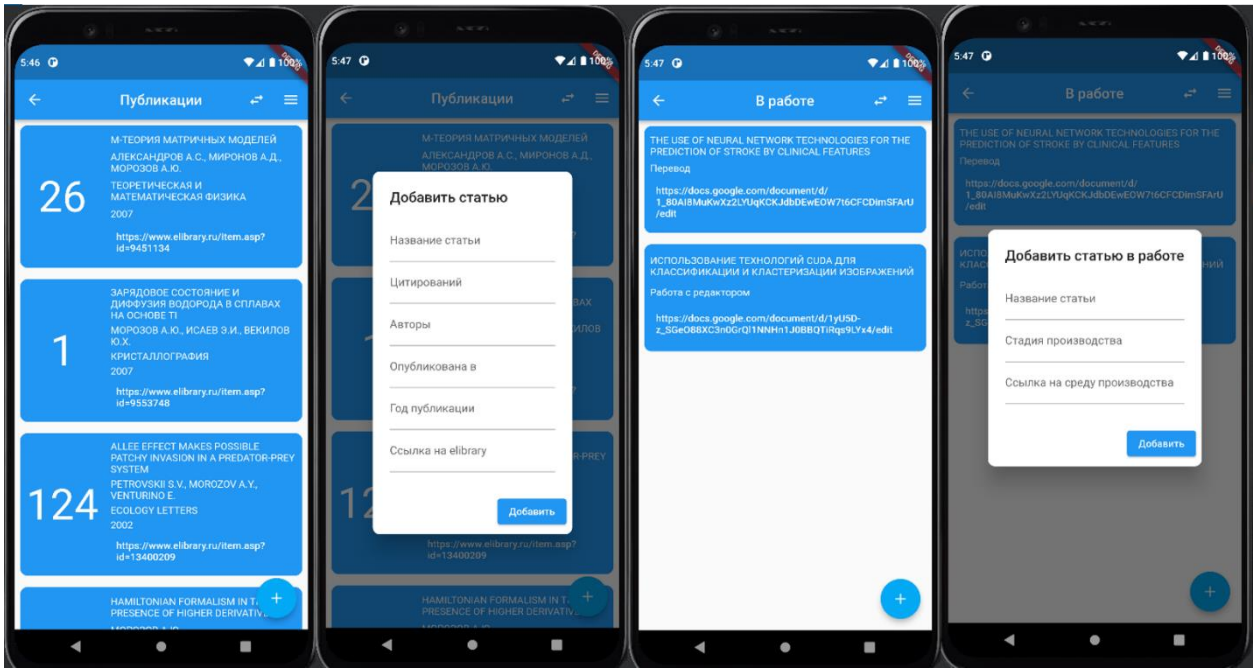

Рисунок 12 – Интерфейс микросервиса «Публикации»

Как видно, микросервис разделён на два экрана, переключение между которыми происходит по нажатию клавиши в верхнем правом углу интерфейса [8].

Каждый из подразделов отображает соответствующий список объектов классов, содержащих информацию о статьях. В подразделах приводится

информация, необходимая для оценки как текущей работы по статьям, так и уже написанных статей.

Также на рисунке 12 приведены текстовые формы для добавления информации о публикациях.

Интерфейс раздела, содержащего информацию о проектной деятельности научного подразделения, имеет общую структуру с разделом о публикациях – рисунок 13 [8].

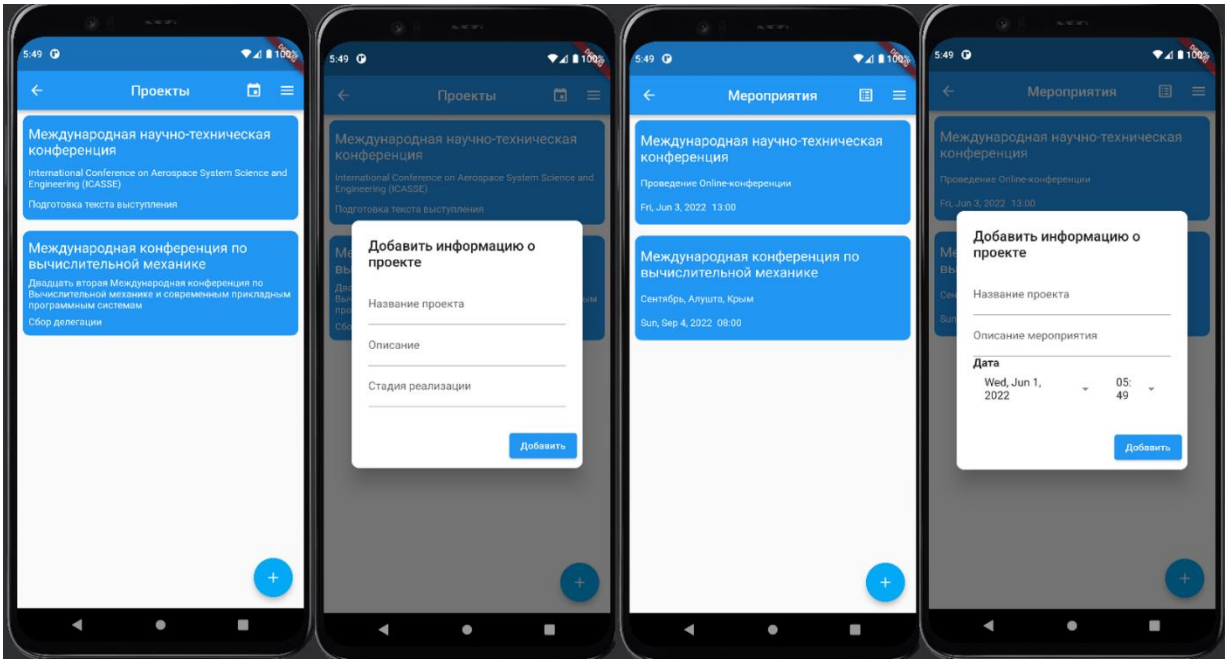

Рисунок 13 – Интерфейс микросервиса «Проекты»

Исключение составляют поля класса мероприятий, хранящие данные в формате DateTime. Способ ввода соответствующих данных в интерфейс отображён на рисунке 14.

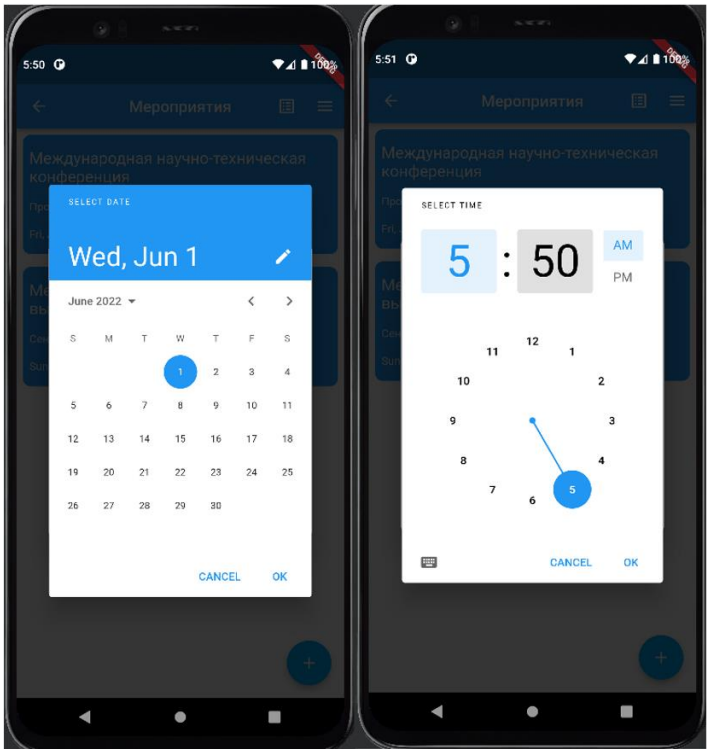

Рисунок 14 – Виджеты showDatePicker и showTimePicker

Микросервис, соответствующий организации работы лабораторного оборудования представлен на рисунке 15.

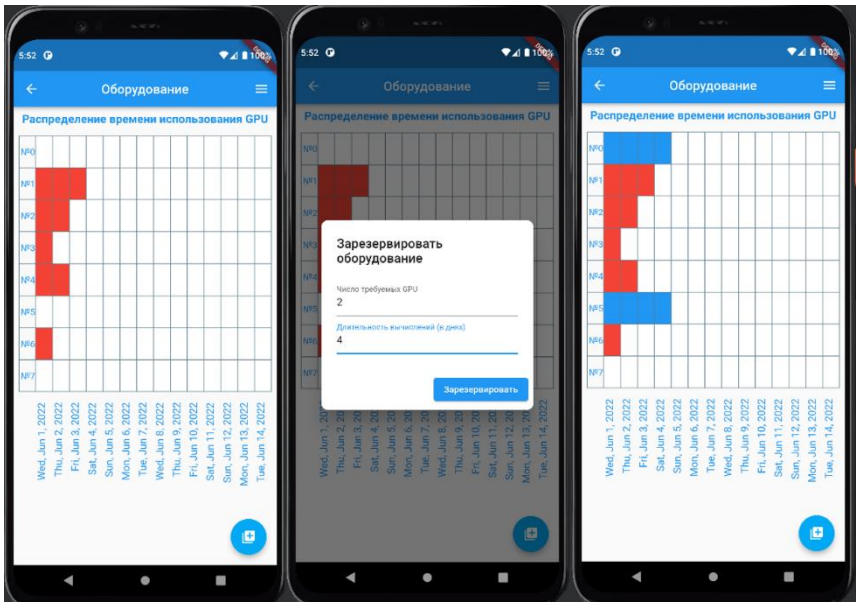

Рисунок 15 - Интерфейс микросервиса «Оборудование»

Основным виджетом раздела является распределение времени работы лабораторного оборудование, представляющее собой таблицу, одной из осей

которой является временной ряд с ценой деления в день, другой – некоторое количество графических устройств. Белым цветом отмечены ячейки с незарезервированными вычислительными мощностями, красным – зарезервированными и синим – зарезервированными текущим пользователем [14].

Также на рисунке 15 приведена форма заявки на использование оборудования.

Интерфейс раздела планирования научной деятельности отображён на рисунке 16.

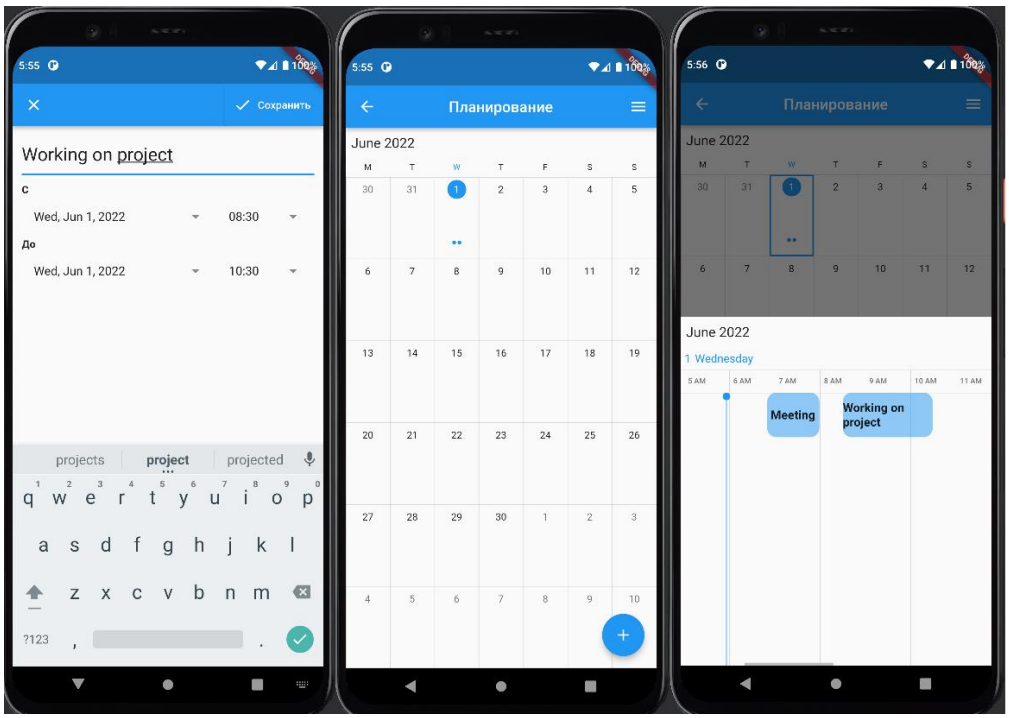

Рисунок 16 - Интерфейс микросервиса «Планирование»

Основа интерфейса данного раздела составлена при помощи пакета syncfusion_flutter_calendar, содержащего в себе виджеты с различными представлениями временного ряда. Так, интерфейс позволяет из календарного

вида перейти в почасовой просмотр временного ряда с расположенными на нём пользователем событиями [9, 14].

Форма для создания нового события, добавляемого во временной ряд реализована при помощи виджета, приведённого на рисунке 14.

На рисунке 17 представлен интерфейс раздела, соответствующего популяризации научной деятельности подразделения.

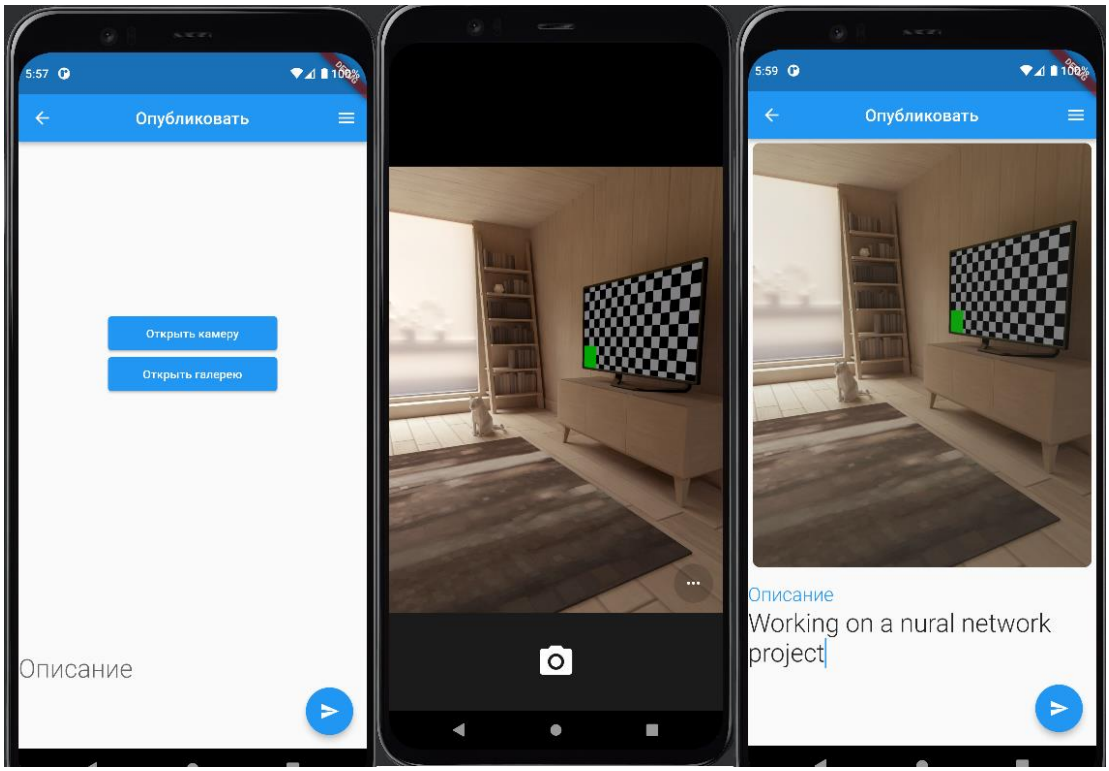

Рисунок 17 - Интерфейс микросервиса «Опубликовать»

Пользователю предлагается выбрать источник медиа-файла, добавить описание. После перечисленных действий публикация готова к отправке на какой-либо информационный ресурс.

Навигация в мобильном приложении осуществлена при помощи меню, отображённого на рисунке 18 [14].

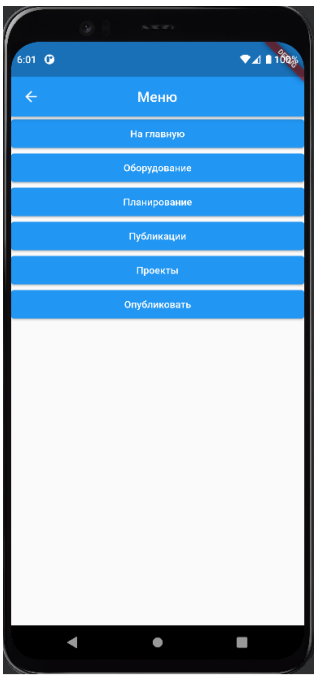

Рисунок 18 – Меню приложения

Таким образом, реализация мобильного приложения содержит в себе все функциональные свойства, описанные в предполагаемом способе решения проблемы.

3.2 Оценка эффективности представленного решения

Благодаря реализации мобильного приложения отсутствует необходимость многократно предоставлять информацию об опубликованных работах или о текущей стадии написания новой публикации. Также, другие сотрудники лаборатории, причастные к написанию определённой работы, теперь имеют быстрый доступ к расположению среды для производства публикации.

Раздел, предоставляющий данные о проектной деятельности, позволяет руководителю научного подразделения оперативно получать информацию о текущем этапе деятельности по каждому из проектов, вести лёгкий учёт будущих мероприятий.

Наличие системы автоматического распределения мощностей лабораторного оборудования позволит существенно сэкономить время и ускорить выполнение вычислительных процессов, совершаемых на его основе [15].

Инструмент для публикации медиа-материалов при должной автоматизации сократит временные траты на размещение информации о деятельности подразделения на различных публикационных площадках.

Грамотное планирование научной деятельности, осуществляемое с участием руководителя подразделение также может значительно оптимизировать рабочий процесс как каждого отдельного сотрудника, так и лаборатории в целом [4].

Также немаловажную роль в экономии временных ресурсов сотрудников играет роль портативности и быстродоступности всех перечисленных инструментов благодаря реализации решения посредством мобильного приложения [1, 8].

3.3 Рекомендации по использованию и развитию разработанного подхода

Первостепенным шагом по развитию мобильного приложения является подключение к микросервису, осуществляющего подготовку медиа-контента к отправке, аккаунтов в различных площадках для публикации такого рода информации.

Также, повышения эффективности разработанного подхода можно добиться, использовав существующую реализацию для подключения Web и настольной версии приложения. Такой шаг значительно повысит доступность

и быстроту взаимодействия с предоставленными в реализации инструментами в различных ситуациях [8, 9].

3.4 Выводы

Исходя из анализа практических результатов реализации приложения для мобильных устройств, можно сделать вывод, что каждый из реализованных в ходе работы микросервисов способен на достаточном уровне сократить временные затраты сотрудника научного подразделения на выполнение различных типовых процессов. Также стоит отметить, что при незначительном улучшении представленного решения, его эффективность может существенно повыситься [4, 6, 13 - 15].

ЗАКЛЮЧЕНИЕ

В выпускной квалификационной работе бакалавра была сформулирована проблема и описан предполагаемый способ её решения с целью сокращения временных затрат сотрудников научной лаборатории на выполнение типовых процессов.

Для достижения вышеуказанной цели были определены типовые рабочие процессы в научной лаборатории, определены направления их совершенствования, разработана архитектура программного решения. На базе разработанной архитектуры выполнена реализация приложения для мобильный устройств, включающее в себя микросервисы, соответствующие типовым рабочим процессам и направленные на снижение временных затрат на выполнение последних.

Разобраны теоретические предпосылки к решению задачи и аспекты практической реализации, представлены графические результаты предложенного решения и произведена оценка его эффективности. Выработаны рекомендации по использованию и развитию разработанного подхода.

Основываясь на полученных результатах, можно сделать вывод, что в ходе выпускной квалификационной работы было создано приложение для мобильных устройств, которое может существенно сократить временные затраты сотрудников научного подразделения института на выполнение типовых научных процессов, а следовательно, цель работы была достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Windmill Eric. Flutter in action // Manning Publications. – 2020.

2 Payne Ray. Beginning App Development with Flutter // Springer. - 2019.

3 Wenhao Wu. React Native vs Flutter, cross-platform mobile application frameworks // Metropolia. - 2018.

4 Kenneth Tobin. Secondary science laboratory activities // Europen Journal of science education. – 1986.

5 Lynn Stewart. Research on science laboratory activities: In pursuit of better questions and answers to improve learning // School science and Mathematics. – 1990.

6 Cerny T., Donahoo M.J., Trnka M., Contextual understanding of microservice architecture: current and future directions // ACM SIGAPP Applied Computing Review, Volume 17, Issue 4. – 2017.

7 Villamizar M., Garcés O., Castro H., Verano M., Salamanca L., Casallas R., Gil S. Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud // 10th Computing Colombian Conference (10CCC) – 2015.

8 Kuzmin N., Ignatiev K., Grafov D., Experience of Developing a Mobile Application Using Flutter // Springer, Singapore. – 2019.

9 Flutter documentation. [электронный ресурс] https://docs.flutter.dev/

10 Dart documentation. [электронный ресурс] https://dart.dev/guides

11 Bracha G. The Dart programming language // Addison-Wesley Professional. – 2015.

12 Warlath K., Ladd S. Dart: Up and Running: A New, Tool-Friendly Language for Structured Web Apps // O'Reilly Media, Inc. – 2012.

13 Bstieler L., Hemmert M. Increasing Learning and Time Efficiency in Interorganizational New Product Development Teams // Journal of Product Innovation. – 2010.

14 Lu J.,, Chen Q., Chen X. App interface study on how to improve user experience // 2012 7th International Conference on Computer Science & Education (ICCSE). – 2012.

15 Шарипов Ф.Ф. Управление учебно-научной лабораторией: новые требования и компетенции // E-Management. – 2020.

16 Khawas C., Shah P., Application of firebase in android app development-a study // International Journal of Computer Applications Volume 179, No.46. – 2018.

ПРИЛОЖЕНИЕ А

Ссылка на репозиторий github с кодом программы: https://github.com/wAlienUFOx/science_lab_2