Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Отчёт по преддипломной практике

Студент А.В. Гуринович

Руководитель В.В. Гуринович

Консультант от кафедры ЭВМ Д.В. Басак

Нормоконтролер Е.Е. Клинцевич

МИНСК 2023

Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

К ЗАЩИТЕ ДОПУСТИТЬ

Зав. каф. ЭВМ

\_\_\_\_\_\_\_\_\_\_ Б.Н. Никульшин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломному проекту

на тему

ПРОГРАММНОЕ СРЕДСТВО ДЛЯ ПРОСМОТРА И

ОБРАБОТКИ ИНФОРМАЦИИ О РАСПИСАНИИ ЗАНЯТИЙ

БГУИР ДП 1–40 02 01 01 019 ПЗ

Студент А.В. Гуринович

Руководитель Д.В. Басак

Консультанты:

от кафедры ЭВМ Д.В. Басак

по экономической части В.Г. Горовой

Нормоконтролер Е.Е. Клинцевич

Рецензент

МИНСК 2023

Реферат

Дипломный проект предоставлен следующим образом. Электронные носители: 1 компакт-диск. Чертёжный материал: 5 листов формата А1. Пояснительная записка: N страниц, N рисунков, N таблиц, N литературных источников, X приложений.

Ключевые слова: Swift, SwiftUI, MVVM, CoreData, Xcode, App Store, адаптивный графические пользовательские интерфейсы, iOS, iPadOS, macOS.

Предметной областью данного проекта является расписание занятий Белорусского государственного университета информатики и радиоэлектроники. Объектом разработки является кроссплатформенное нативное приложение для просмотра и обработки информации о расписании занятий Белорусского государственного университета информатики и радиоэлектроники.

Целью разработки данного дипломного проекта является создание приложения с высокой степенью удобства и быстродействия, которое будет использовать всю доступную через интерфейс программирования приложений интегрированной информационной системы Белорусского государственного университета информатики и радиоэлектроники информацию для отображения расписания, и иной информации.

Для разработки данного проект использовалась интегрированная среда разработки Xcode, язык программирования Swift и платформа SwiftUI для создания пользовательского интерфейса.

В результате разработки создано программное средство для просмотра и обработки информации о расписании занятий для операционных систем iOS, iPadOS и macOS. Программное средство доступно для загрузки на платформе Apple App Store.

Практическим применением разработки является удобное и быстрое получение структурированной информации для студентов и сотрудников университета. Пользователи смогут легко находить информацию о своих занятиях, преподавателях, группах, кабинетах, а также добавлять задания со сроком выполнения и управлять ими. Проект улучшает опыт обучения в университете для пользователей на платформах Apple, снижает время, затрачиваемое на получение и обработку информации.

Проект является эффективным с экономической стороны, при этом эффективно выражается не только в прямой прибыли от продаж, но и в повышении эффективности пользователей в процессе обучения.

Данные проект полностью завершён и доступен пользователям на рынке, в частности, в Apple App Store. Дальнейшему развитию приложения может поспособствовать открытый исходный код, который доступен на web-сервисе GitHub, где другие пользователи могут сообщать об ошибках, предлагать и реализовывать улучшения программного средства.

Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет: КСиС. Кафедра: ЭВМ.

Специальность: 40 02 01 «Вычислительные машины, системы и сети».

Специализация: 40 02 01-01 «Проектирование и применение локальных компьютерных сетей».

УТВЕРЖДАЮ

Заведующий кафедрой ЭВМ

\_\_\_\_\_\_\_\_\_ Б.В.Никульшин

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 г.

ЗАДАНИЕ

по дипломному проекту студента

Гуриновича Андрея Викторовича

1 Тема проекта: «Программное средство для просмотра и обработки информации о расписании занятий» – утверждена приказом по университету от 24 марта 2023 г. № 743-с.

**2** Срок сдачи студентом законченного проекта: 1 июня 2023 г.

**3** Исходные данные к проекту:

1. Получение данных из интегрированной информационной системы Белорусского государственного университета информатики и радиоэлектроники.
2. Операционные системы: iOS, iPadOS и macOS.
3. Среда разработки: Xcode.
4. Язык программирования: Swift.

**4** Содержание пояснительной записки (перечень подлежащих разработкевопросов):

Введение 1. Обзор литературы. 2. Системное проектирование. 3. Функциональное проектирование. 4. Разработка программных модулей. 5. Программа и методика испытаний. 6. Руководство пользователя. 7. Технико-экономическое обоснование разработки и реализации на рынке программного средства для просмотра и обработки информации о расписании занятий. Заключение. Список использованных источников. Приложения.

**5** Перечень графического материала (с точным указанием обязательных чертежей):

1. Вводный плакат. Плакат.
2. Заключительный плакат. Плакат.
3. Программное средство для просмотра и обработки информации о расписании занятий. Схема структурная.
4. Программное средство для просмотра и обработки информации о расписании занятий. Модель данных.
5. Программное средство для просмотра и обработки информации о расписании занятий. Диаграмма классов.
6. Программное средство для просмотра и обработки информации о расписании занятий. Диаграмма последовательностей.
7. Программное средство для просмотра и обработки информации о расписании занятий. Схема программы.

**6** Содержание задания по экономической части: «Технико-экономическое обоснование разработки и реализации на рынке средства для просмотра и обработки информации о расписании занятий».

ЗАДАНИЕ ВЫДАЛ \_\_\_\_\_\_\_\_ В.Г. Горовой

КАЛЕНДАРНЫЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапов дипломного проекта | Объем этапа, % | Срок выполнения этапа | Примечания |
| Подбор и изучение литературы | 10 | 23.03 – 29.03 |  |
| Структурное проектирование | 10 | 29.04 – 05.04 |  |
| Функциональное проектирование | 20 | 05.04 – 18.04 |  |
| Разработка программных модулей | 30 | 18.04 – 08.05 |  |
| Программа и методика испытаний | 10 | 08.05 – 11.05 |  |
| Расчёт экономической эффективности | 10 | 11.05 – 15.05 |  |
| Оформление пояснительной записки | 10 | 15.05 – 21.05 |  |

Дата выдачи задания: 23 марта 2023 г.

Руководитель \_\_\_\_\_\_\_\_ Д.В. Басак

ЗАДАНИЕ ПРИНЯЛ К ИСПОЛНЕНИЮ \_\_\_\_\_\_\_\_ А.В. Гуринович

Содержание

[ВВЕДЕНИЕ 8](#_Toc133885767)

[1 Обзор литературы 9](#_Toc133885768)

[1.1 Интерфейс программирования приложений интегрированной информационной системы Белорусского государственного университета информатики и радиоэлектроники 9](#_Toc133885769)

[1.2 Выбор платформы для разработки 14](#_Toc133885770)

[1.3 Интегрированная среда разработки Xcode 19](#_Toc133885771)

[1.4 Обзор аналогов 20](#_Toc133885772)

[2 Системное проектирование 24](#_Toc133885773)

[2.1 Модели 24](#_Toc133885774)

[2.2 Представления 25](#_Toc133885775)

[2.3 Модели Представлений 25](#_Toc133885776)

[2.4 Блок извлечения 26](#_Toc133885777)

[2.5 Блок декодирования 26](#_Toc133885778)

[2.6 Постоянное хранилище 26](#_Toc133885779)

[2.7 Фоновый контекст 27](#_Toc133885780)

[2.8 Контекст Представления 27](#_Toc133885781)

[3 Функциональное проектирование 28](#_Toc133885782)

[3.1 Модели 28](#_Toc133885783)

[3.2 Представления 28](#_Toc133885784)

[3.3 Модели Представлений 28](#_Toc133885785)

[3.4 Блок извлечения 28](#_Toc133885786)

[3.5 Блок декодирования 28](#_Toc133885787)

[3.6 Постоянное хранилище 29](#_Toc133885788)

[3.7 Фоновый контекст 29](#_Toc133885789)

[3.8 Контекст Представления 29](#_Toc133885790)

[4 Разработка программных модулей 30](#_Toc133885791)

[5 Программа и методика испытаний 31](#_Toc133885792)

[6 Руководство пользователя 32](#_Toc133885793)

[7 Технико-экономическое обоснование разработки и реализации на рынке ПРОГРАММНОГО средства для просмотра и обработки информации о расписании занятий 33](#_Toc133885794)

[7.1 Характеристика программного средства, разрабатываемого для реализации на рынке 33](#_Toc133885795)

[7.2 Расчёт инвестиций в разработку программного средства 33](#_Toc133885796)

[7.3 Расчёт экономического эффекта от реализации программного средства на рынке 36](#_Toc133885797)

[7.4 Расчёт показателей экономической эффективности разработки и реализации программного средства на рынке 38](#_Toc133885798)

[7.5 Вывод об экономической целесообразности реализации проектного решения 39](#_Toc133885799)

[Заключение 40](#_Toc133885800)

[Список использованных источников 41](#_Toc133885801)

[ПРИЛОЖЕНИЕ А 43](#_Toc133885802)

[ПРИЛОЖЕНИЕ Б 44](#_Toc133885803)

ВВЕДЕНИЕ

Данный дипломный проект посвящён разработке программного средство для просмотра и обработки информации о расписании занятий Белорусского государственного университета информатики и радиоэлектроники. Разработка программного средства для разных типов устройству требует…

Адаптация к существующему API

Упор на обработку информации, а не отображение

Во введении кратко указывается, чему посвящён дипломный проект, делается обзор состояния предметной области, формулируется общая цель разработки и её место среди известных отечественных и зарубежных аналогов.

# Обзор литературы

## Интерфейс программирования приложений интегрированной информационной системы Белорусского государственного университета информатики и радиоэлектроники

Интерфейс программирования приложений (далее – API) интегрированной информационной системы (далее – ИИС) Белорусского государственного университета информатики и радиоэлектроники (далее – БГУИР) предоставляет программные интерфейсы, предназначенные для получения информации о расписании групп и преподавателей, а также информацию о факультетах, кафедрах, специальностях и аудиториях [[[1]](#endnote-2)].

API имеет возможность предоставления ответа на запрос как в виде JavaScript Object Notation (далее – JSON), так и в виде так и в виде Extensible Markup Language (далее – XML). Так как Swift имеет встроенную библиотеку для расшифровки JSON, в разработке проекта будет использованы ответы в нотации JSON. Для XML потребовалось бы использовать сторонний модуль расшифровки, либо разрабатывать его самостоятельно.

Далее в данном подразделе представлены структуры ответов API ИИС на различные запросы. В каждой структуре приведено названия ключей и типы значений. В случае, если значение типа является опциональным, оно приводится со знаком вопроса в конце названия, это соответствует нотации обозначения опциональных типов в языке программирования Swift. Полями, в данном подразделе, далее будут называться пары ключа и значения.

### Факультет

Структура представлена в таблице:

Таблица 1.1 – Структура факультета из API ИИС

|  |  |  |
| --- | --- | --- |
| Ключ | Тип | Описание |
| id | Число | Идентификатор |
| name | Строка | Название |
| abbrev | Строка | Аббревиатура |

### Специальность

Некоторые специальности содержат идентификаторы факультета, которые не сопоставляются ни с одним из получаемых запросом факультетом объектов. Такое поведение вызвано тем, что в ответе на запрос специальностей содержатся также те специальности, обучение по которым более не производится, либо производится через Систему электронного обучения БГУИР. Также существуют специальности, которые ссылаются на более несуществующие факультеты, либо на их предыдущие идентификаторы.

Структура представлена в таблице:

Таблица 1.2 – Структура специальности из API ИИС

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| id | Число | Идентификатор |
| name | Строка | Название |
| abbrev | Строка | Аббревиатура |
| code | Строка | Код |
| facultyId | Число | Идентификатор факультета |

### Кафедра

Ответ на данный запрос предоставляет только кафедры, другие подразделения, к которым, могут относится кабинеты, в нём не содержатся.

Структура представлена в таблице:

Таблица 1.3 – Структура кафедры из API ИИС

|  |  |  |
| --- | --- | --- |
| Ключ | Тип | Описание |
| id | Число | Идентификатор |
| name | Строка | Название |
| abbrev | Строка | Аббревиатура |

### Аудитория

Аудитория имеет две вложенных структуры: тип аудитории и номер здания. Типы аудитории принимают одно из следующих значений: лекционные занятия, практические занятия, лабораторные занятия, компьютерный класс.

Структура представлена в таблице:

Таблица 1.4 – Структура типа аудитории из API ИИС

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| id | Число | Идентификатор |
| name | Строка | Название | |
| abbrev | Строка | Аббревиатура | |

Номер здания представляет из себя структуру, в которой содержится только идентификатор и название здания. При этом идентификатор не совпадает с номером корпуса, это актуально только для корпусов с первого по пятый, поэтому значащим полем в первую очередь является название здания.

Даже аудитории, которые находятся вне университета, такие как, например, находящиеся в зданиях EPAM Systems в Парке высоких технологий, имеют присвоенное здание, в котором они де-факто не находятся. Например, вышеупомянутые кабинетам присвоен второй корпус.

Структура представлена в таблице:

Таблица 1.5 – Структура номера здания из API ИИС

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| id | Число | Идентификатор |
| name | Строка | Название |

Значение вместимости аудитории встречается очень редко, чаще всего у компьютерных классов, только 27 аудиторий имеют значение для этого ключа. Аудитории также могут не иметь присвоенной кафедры.

Структура представлена в таблице:

Таблица 1.6 – Структура аудитории из API ИИС

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| id | Число | Идентификатор |
| name | Строка | Название |
| note | Строка? | Заметка |
| capacity | Число? | Вместительность |
| department | Кафедра? | В таблице 1.3 |
| auditoryType | Тип аудитории | В таблице 1.4 |
| buildingNumber | Номер здания | В таблице 1.5 |

### Группа

В случае с группой и преподавателем существуют некоторые особенности, данные структуры предоставляются в двух представлениях: обычном, которое является ответом на запрос всех групп и расписания группы, и встроенном, которое является частью структуры занятия. При этом они имеют как общие поля, которые содержатся в обоих представлениях, так и содержащиеся исключительно в одном представлении поля.

Структура представлена в таблице:

Таблица 1.7 – Структура группы из API ИИС

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип | Представление | | Описание |
| Обычное | Встроенное |
| 1 | 2 | 3 | 4 | 5 |
| id | Число | • |  | Идентификатор |
| name | Строка | • | • | Номер |
| calendarId | Строка | • |  | Идентификатор календаря Google Calendar |

Продолжение таблицы 1.7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| course | Число | • |  | Курс |
| educationDegree | Число | • | • | Степень образования |
| numberOfStudents | Число |  | • | Количество студентов |
| facultyId | Строка | • |  | Идентификатор факультета |
| facultyAbbrev | Строка | • |  | Аббревиатура факультета |
| specialityName | Строка | • | • | Название специальности |
| specialityAbbrev | Строка | • |  | Аббревиатура специальности |
| specialityCode | Строка |  | • | Код специальности |
| specialityDepartment  EducationFormId | Число | • |  | Идентификатор специальности |

### Преподаватель

Структура преподавателя, имеет схожие со структурой группы особенности, однако в ней меньше непересекающихся полей, а встроенное представление вовсе содержит все поля.

Значение степени в обычном представлении имеет значение аббревиатуры степени, а в встроенном представлении же имеет значение полного названия степени.

Структура представлена в таблице:

Таблица 1.8 – Структура преподавателя из API ИИС

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип | Представление | | Описание |
| Обычное | Встроенное |
| 1 | 2 | 3 | 4 | 5 |
| id | Строка | • | • | Идентификатор |
| firstName | Строка | • | • | Имя |
| middleName | Строка? | • | • | Отчество |
| lastName | Строка | • | • | Фамилия |
| rank | Строка? | • | • | Ранг |
| degree | Строка? | • | • | Степень либо аббревиатура |
| degreeAbbrev | Строка? |  | • | Аббревиатура степени |

Продолжение таблицы 1.8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| urlId | Строка | • | • | Идентификатор для доступа через сайт |
| calendarId | Строка | • | • | Идентификатор календаря Google Calendar |
| photoLink | Строка | • | • | Ссылка на фотографию |
| academicDepartment | Строки | • | • | Список аббревиатур кафедр |
| email | Строка? |  | • | Адрес электронной почты |
| jobPositions | Строки? |  | • | Должности |

### Расписание

Структуры группы и преподавателя существуют только для запроса соответствующего расписания. Данные структуры не могут встретиться в одном расписании вместе. Так как в структуре занятия отсутствует информация о дне недели, от предоставляется в контейнере занятий по дням недели. Занятия по дням недели включают экзамены только в том случае, если существуют обычные занятия.

Структура представлена в таблице:

Таблица 1.9 – Структура расписания из API ИИС

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| schedules | Занятия по дням недели | Словарь, где ключом является день недели, а значением является массив занятий |
| exams | Занятия | В таблице 1.10 |
| startDate | Строка? | Уникальный идентификатор аудитории |
| endDate | Строка? | Название аудитории |
| startExamsDate | Строка? | Заметка |
| endExamsDate | Строка? | Вместимость аудитории, очень редко обозначена |
| studentGroupDto | Группа | В таблице 1.7 |
| employeeDto | Преподаватель | В таблице 1.8 |

### Занятие

Структура занятия является наиболее сложной, так как содержит много опциональных значений, имеет в себе другие структуры, которые ссылаются далее на третье структуры. Именно занятие связывает аудитории, группы и преподавателей.

Опциональность аудиторий, групп и преподавателей обусловлена тем, что при запросах расписания группы, для занятия может не быть назначен преподаватель. Например, если преподаватель не является сотрудником университета, а занятия проходят на базе предприятия под руководством его сотрудников.

Аналогично и занятия со стороны расписания преподавателя могут быть временно не назначены группы.

Структура представлена в таблице:

Таблица 1.10 – Структура занятия из API ИИС

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| subject | Строка | Аббревиатура названия дисциплины |
| subjectFullName | Строка | Название дисциплины |
| weekNumber | Числа | Список номеров недель, |
| startLessonTime | Строка | Время начала занятия |
| endLessonTime | Строка | Время окончания |
| numSubgroup | Число | Номер подгруппы |
| split | Булево значение | Определяет, является ли занятие разделённым |
| note | Строка? | Заметка |
| lessonTypeAbbrev | Строка | Аббревиатура типа занятия |
| announcement | Булево значение | Определяет, является ли это занятие объявлением |
| dateLesson | Строка? | Дата занятия |
| startLessonDate | Строка | Дата начала периода проведения данного занятия |
| endLessonDate | Строка | Дата окончания периода проведения данного занятия |
| auditories | Строки? | Названия аудиторий и здание |
| studentGroups | Группы? | Список групп, |
| employees | Преподаватели? | Список преподавателей |

## Выбор платформы для разработки

Создание приложений на языке программирования Swift в первую очередь подразумевает использование трёх платформ: UIKit для iOS и iPadOS или AppKit для macOS и SwiftUI для всех операционных систем (далее – ОС). Они используют соответственно паттерны Model-View-Controller и View-ViewModel-Model.

### Model-View-Controller

Model-View-Controller (далее – MVC) является паттерном, который разделяет приложение Модель (Model), Представление (View) и Контроллер (Controller) [[[2]](#endnote-3)]. Структура паттерна представлена на рисунке:



Рисунок 1.1 – схема структуры паттерна MVC

Модель отвечает за хранение данных и логику обработки данных: в ней описывается бизнес-логика, правила валидации и другие операции, связанные с данными. Модель не имеет информации о Представлением и Контроллером, то есть не имеет доступа к их полям, функциям и иному.

Представление отображает данные пользователю и предоставляет возможность взаимодействия с приложением через графический интерфейс. В ней определяется внешний вид и макет интерфейса, это происходит в графическом редакторе. Представление не имеет представления о существовании Модели и Контроллера.

Контроллер обрабатывает пользовательский ввод и управляет взаимодействием Модели и Представления. В ней содержится логика приложения, отвечающая за обработку событий, получаемых от Представления, и изменение Модели соответствующим образом. Контроллер имеет исходящую связь с Представлением или Моделью.

### UIKit и AppKit

UIKit является основной платформой для создания пользовательских интерфейсов на iOS и iPadOS [[[3]](#endnote-4)]. В платформе применяется в первую очередь императивное программирование. UiKit представляет собой набор готовых компонентов интерфейса, таких как кнопки, текстовые поля, таблицы, контейнеры, иные элементы интерфейса и контейнеры для них. SwiftUI паттерн проектирования архитектуры MVC.

В целом, UIKit и SwiftUI представляют собой две разные платформы для создания пользовательских интерфейсов в операционных системах Apple, каждая со своими плюсами и минусами. Разработчики должны выбирать платформу, которая лучше всего подходит для их задач и опыта в программировании.

AppKit является платформой для создания приложений под macOS [[[4]](#endnote-5)]. Платформа является набором инструментов и ресурсов, который разработчики могут использовать для создания графических пользовательских интерфейсов.

UIKit и AppKit имеют следующую парадигму для создания пользовательских интерфейсов: используются файлы xib или Storyboard, в которых определяются все элементы интерфейса и их свойства, а уже в файлах Swift программируется поведение и особенности интерфейса, программный код в данном случае пишется в императивной парадигме. Обе платформы используют паттерн Model-View-Controller.

AppKit и UIKit очень похожи и имеют одинаковую структуру и логику. Элементы имеют одинаковые названия и одинаковые параметры, хоть и существуют особенные для платформ элементы. Несмотря на это, эти платформы не являются взаимозаменяемыми, хоть технически и существует возможность создания приложения сразу на двух платформах, по сути это два разных приложения, которые объединены только Моделью, а все элементы графического интерфейса необходимо имплементировать отдельно для каждой из платформ.

### Model-View-ViewModel

Model-View-ViewModel (далее – MVVM) является схожим с MVC паттерном, используемым для разделения компонентов приложения на три основных уровня: Модель, Представление и Модель Представления (ViewModel). Структура паттерна представлена на рисунке:



Рисунок 1.2 – схема структуры паттерна MVVM

Модель, аналогична Модели MVC, не имеет информации о Представлением и Модели Представления.

Представление, частично похоже на MVC, отображает данные пользователю и обрабатывает пользовательский ввод. При это в отличии от MVC, представление напрямую обращается к Модели Представления, последняя чаще всего является полем Представления.

Модель Представления является посредником между моделью и представлением и чем-то схожа с Контроллером MVC. Модель имеет общее с Представляем данные, которые автоматически обновляются и Модели Представления и в самом Представлении при изменении их как стороны пользователя, так и со стороны модели. В MVC же Контроллер при обновлении данных запускает метод обновления Представления. В каком-то смысле Представление совмещает Контроллер и Представление MVC.

Одним из недостатков MVVM является некоторая его избыточность для простых приложений, которые имеют небольшое количеством логики и данных. MVVM требует больше кода, чем, например, простая парадигма Model-View-Controller. При этом эти недостатки нивелируются при разработке относительно больших приложений, так как MVVM предоставляет огромные возможности для повторного использования различных элементов Представления. На начальных этапах разработка приложение, использующего платформу MVVM, может быть затратным с точки зрения количества кода и времени разработки, однако повторное использование и универсальность стандартизированных представлений.

### SwiftUI

SwiftUI является более современной платформа для создания пользовательских интерфейсов, которая была запущена в 2019 году [[[5]](#endnote-6)]. В платформе применяется в первую очередь декларативное программирование. SwiftUI основан на декларативном подходе, где для создания интерфейса описывается его конечное представление в виде дерева вложенных представлений. SwiftUI паттерн проектирования архитектуры MVVM.

Одним из основных преимуществ SwiftUI является более простой и интуитивный синтаксис для описания пользовательского интерфейса. Кроме того, SwiftUI обладает рядом функций, которых нет в UIKit, таких как автоматические анимации и адаптивный дизайн, что упрощает создание интерфейсов, которые могут адаптироваться к разным размерам экранов.

Также преимуществом SwiftUI является возможность непосредственной работы с данными в представлениях, благодаря связыванию данных (Data Binding), который позволяет отслеживать изменения данных в Модели или Модели Представления и автоматически обновлять Представление, что значительно упрощает разработку приложений.

На данный момент SwiftUI не поддерживает все функции, которые доступны в UIKit, для SwiftUI 4.0 чаще всего это узкоспециализированные возможности. Для добавления элементов из UIKit в структуру SwiftUI существует структура UIViewControllerRepresentable [[[6]](#endnote-7)], которая позволяет взаимодействовать с элементами UIKit и преобразовывать их в элементы путём некоторых манипуляций SwiftUI.

### Сравнение UIKit и AppKit с SwiftUI в контексте разрабатываемого программного средства

Парадигма программирования: UIKit использует императивный подход к программированию, который требует явного указания всех деталей в коде, в то время как SwiftUI использует декларативный подход, который позволяет описывать только конечный результат, а не метод его достижения. Декларативная парадигма является более удобной для разработчика, однако может усложнить разработку при создании крайне специализированных элементов интерфейсов. Так как разрабатываемое программное средство не подразумевает создания таких элементов, SwiftUI является более подходящим для разработки.

Среди исходных данных к дипломному проекту приведена кроссплатформенность, а именно реализация приложения для iOS, iPadOS и macOS. Разработка кроссплатформенного приложения UIKit и AppKit потребует большего количества времени, так как Представление и Контроллер необходимо разрабатывать отдельно для каждой платформы. Хоть это и были бы крайне схожие элементы, необходимость параллельной разработки значительно затруднила бы какие-либо изменения проекта. Однако, среди преимуществ UIKit и AppKit можно выделить возможность создания более продвинуты и эстетически нестандартных пользовательских графических интерфейсов.

SwiftUI обеспечивает адаптивный дизайн, который автоматически изменяет интерфейс в зависимости от размера Представления, а UIKit не имеет встроенной поддержки адаптивности, то есть требует отдельных расчётов для размеров элементов интерфейса разных устройств, что потребует большого количества времени в контексте разработки данного программного средства для разных типов устройств с разными размерами дисплеев.

С точки сторонних библиотек и инструментов: UIKit имеет более широкий выбор сторонних библиотек и инструментов для разработки, в то время как для SwiftUI являются более ограниченными.

Так как SwiftUI имеет декларативною парадигмы программирования, автоматический адаптивный дизайн и возможности создания кроссплатформенных приложений для iOS, iPadOS и macOS без необходимости параллельной разработки Представления и Контроллера для каждой платформ, платформа оптимальной для разработки данного программного средства.

## Интегрированная среда разработки Xcode

Xcode является интегрированной средой разработки (далее – IDE) для приложений, предназначенных для платформ Apple: iOS, iPadOS, macOS, watchOS и tvOS. Xcode позволяет создавать приложения, тестировать их, отлаживать и развёртывать.

Xcode включает в себя различные редакторы кода, визуальные редакторы пользовательских интерфейсов, отладчики и профилировщики производительности, а также инструменты для автоматизации и управления проектами.

Возможно проводить установку и отладку приложений как в симуляторе, так и на устройстве, которое можно подключить как проводным, так и беспроводным образом. В режиме отладки Xcode выводит подробную информацию о работе устройства или симулятора. Например, возможно приостановить выполнение приложения и рассмотреть отображаемый графический пользовательский интерфейс как набор слоёв.

Xcode также включает в себя библиотеку функций, которые могут быть использованы для создания приложений: инструменты для работы с базами данных, мультимедийными файлами, сетевыми протоколами, графикой.

Мгновенный просмотр результатов компиляции графического пользовательского интерфейса, разработанного на платформе SwiftUI, позволяет мгновенно отображать последствия изменения кода реальном времени. Это разработчикам оперативно тестировать новые идеи и экспериментировать с различными вариантами оформления, не теряя время на компиляцию и запуск приложения в симуляторе или на устройстве. Также доступен мгновенный просмотр сразу для нескольких вариантов отображения: разные макеты устройств, ориентации, цветовые темы и иные модификации.

## Обзор аналогов

На момент написания дипломной работы существуют только два схожих программных средства для iOS; одно – для iPadOS. Они распространяются в Apple App Store. Для macOS приложений, аналогичных разрабатываемому, не существует. Представленные версии приложений для не являются нативными для macOS и запускаются через системную прослойку. В пунктах приведён обзор существующих приложений, их основные характеристики, возможности, ограничения.

### BSUIR Timetable

BSUIR Timetable представляет из себя приложение доступное для iOS версии 13,0 и выше и macOS 11,0 и выше с чипом Apple M1 и выше. [[[7]](#endnote-8)] Приложение разработано на платформе UIKit. Средняя оценка приложения на основании 32 отзывов составляет 3,7 балла. Поддерживаются русский и английский языки. Монетизация происходит с использованием рекламы и встроенной покупки для её отключения. Стоимость данной покупки составляет 0,99 Долларов США.

Основные возможности приложения представлены в таблице:

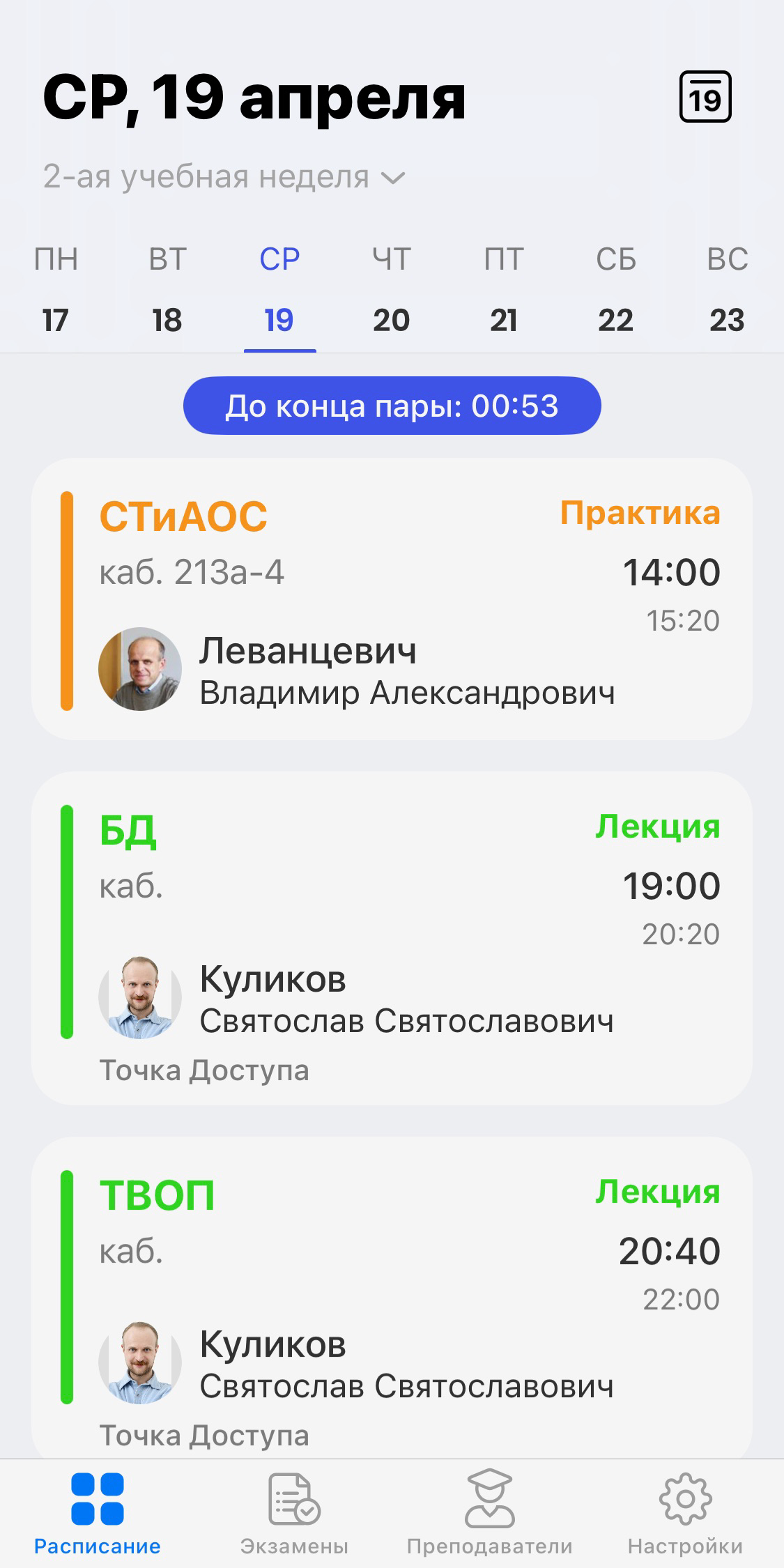
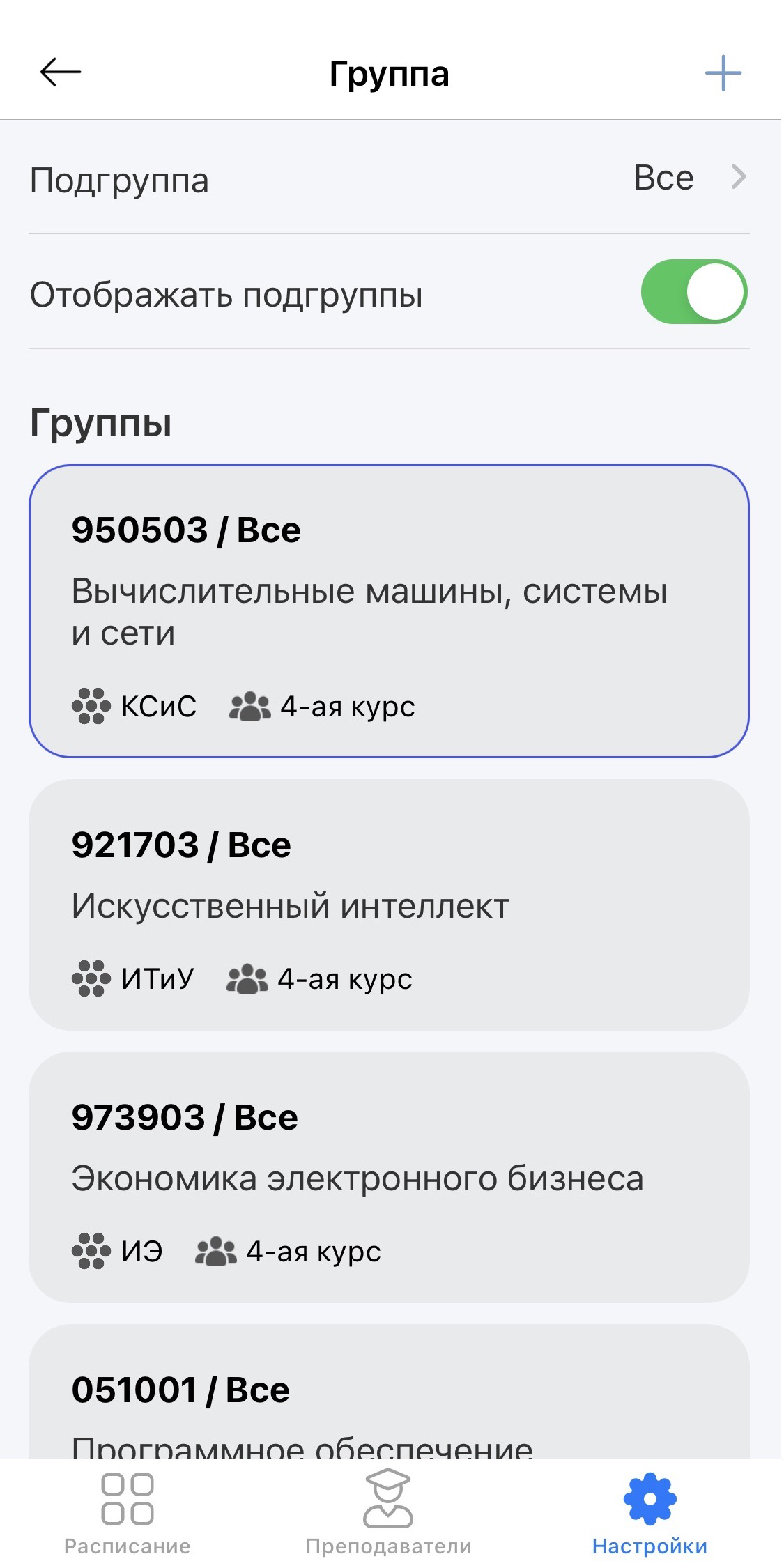
Таблица 1.11 – Основные возможности BSUIR Timetable

|  |  |
| --- | --- |
| Секция | Возможность |
| 1 | 2 |
| Группа | Поиск по номеру |
| Отображение списка всех групп |
| Отображение расписания |
| Отображение факультета |
| Отображение специальности |
| Выбор подгруппы |
| Добавление в избранные |

Продолжение таблицы 1.11

|  |  |
| --- | --- |
| 1 | 2 |
| Преподаватель | Поиск по фамилии |
| Отображение расписания |
| Отображение списка всех преподавателей |
| Отображение списка текущих преподавателей |
| Расписание | Отображение по датам |
| Отображение по неделям |
| Отображение времени до занятия |
| Уведомления о начале занятия |
| Виджеты |

Интерфейс данного приложения представлен на рисунке:

**

*а) б)*

Рисунок 1.3 – интерфейс приложения BSUIR Timetable:

*а* – расписание группы; *б* – выбор из избранных групп

### Bsuir Schedule

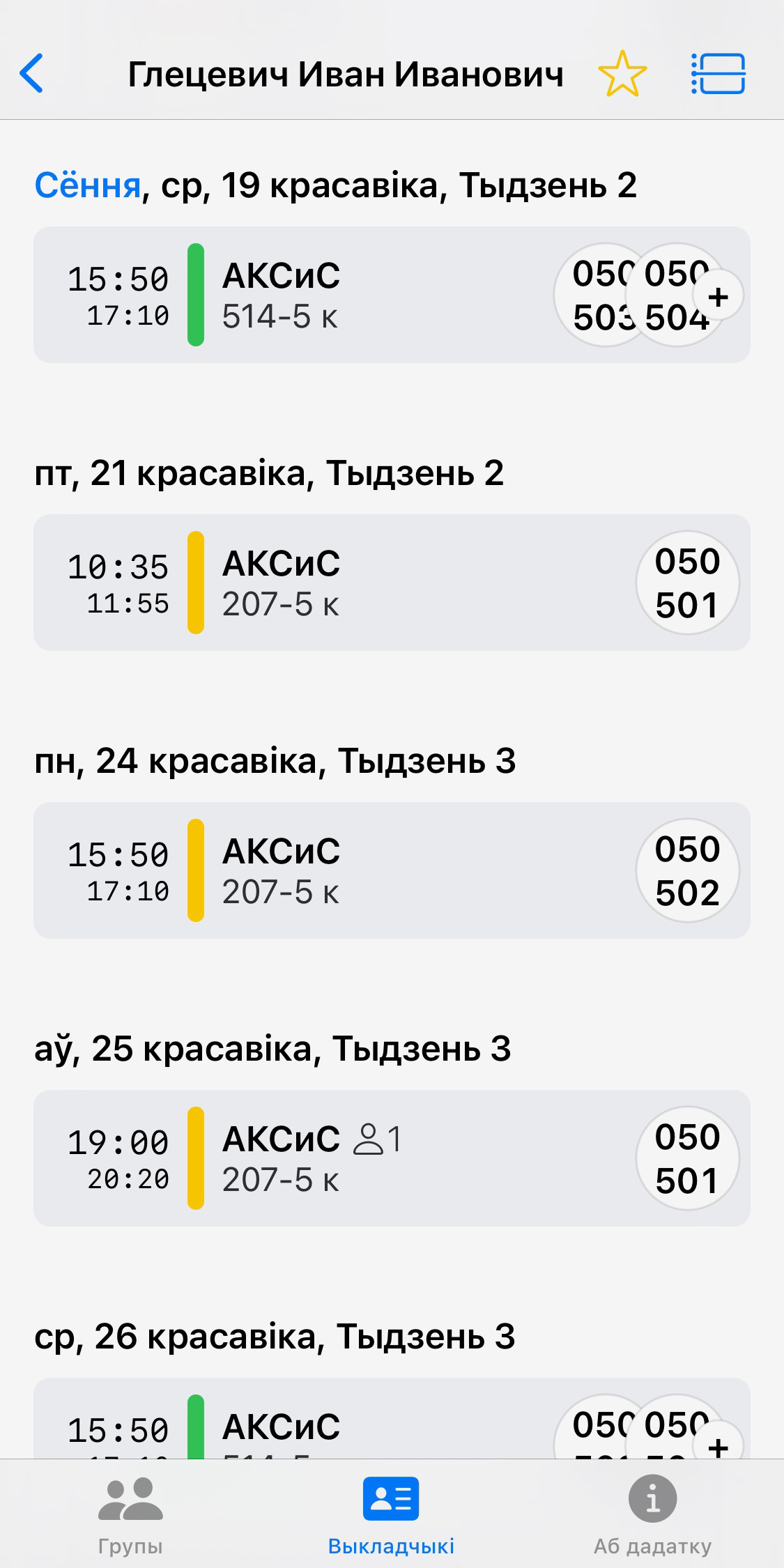
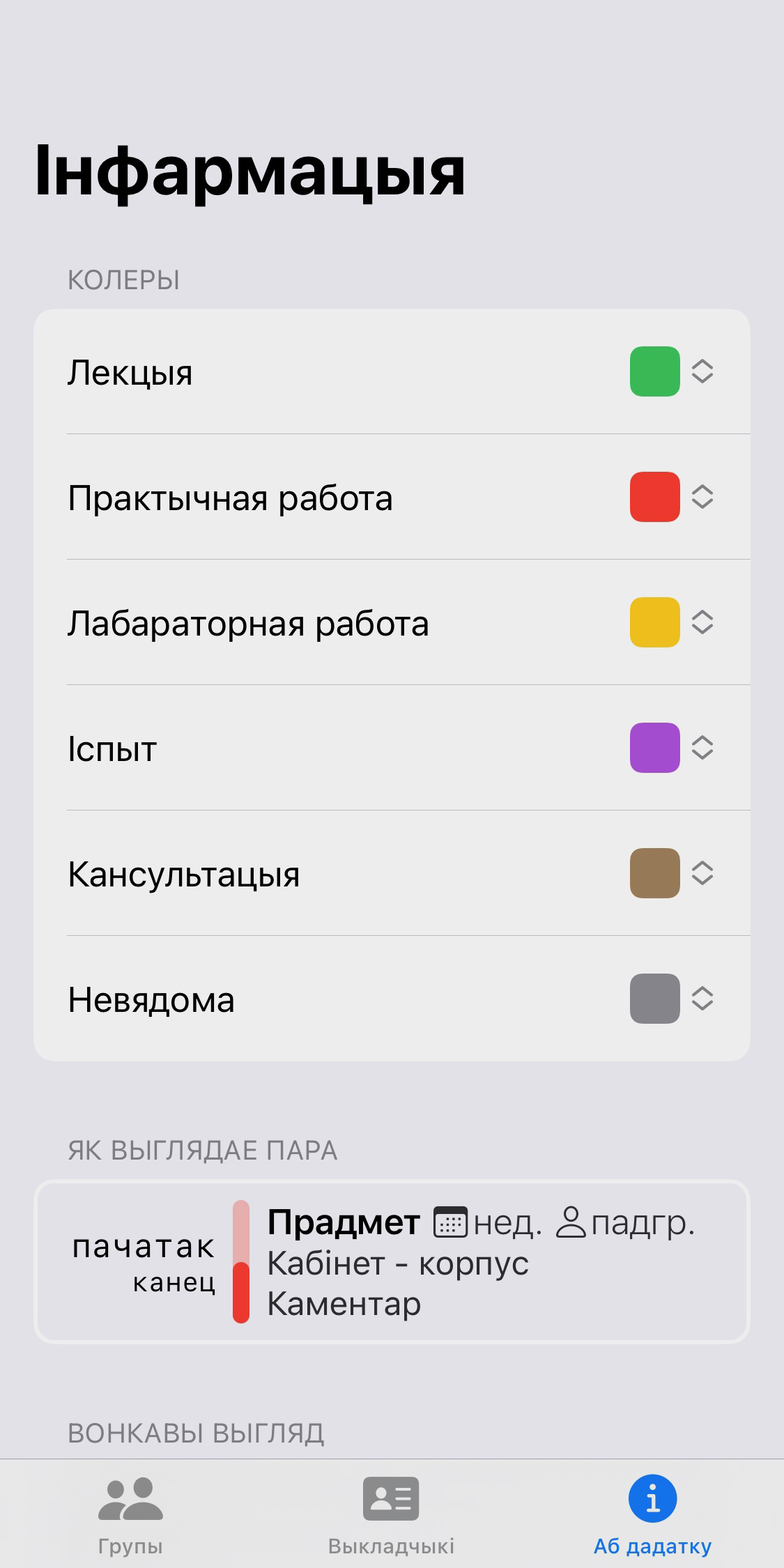
Bsuir Schedule является приложением, которое доступно для iOS и iPadOS версии 15,0 и выше и macOS 11,0 и выше с чипом Apple M1 и выше [[[8]](#endnote-9)]. Приложение разработано на платформе SwiftUI. Средняя оценка приложения на основании 409 отзывов составляет 4,6 балла. Поддерживаются белорусский, украинский, русский и английский языки. Монетизация отсутствует, приложение распространяется на полностью безвозмездной основе. Исходные коды приложения доступны на платформе. GitHub [[[9]](#endnote-10)].

Данное приложение является наиболее продвинутым среди аналогов, однако его архитектура предполагает в перовую очередь просмотр расписания занятий, а не его обработку. Данное приложение не хранит информацию о расписании, она сбрасывается при каждом выходе из приложения. Разрабатываемое же в данном дипломном проекте приложение подразумевает создание продвинутого приложения, упор в котором сделан на обработку информации о расписании.

Таблица 1.12 – Основные возможности Bsuir Schedule

|  |  |
| --- | --- |
| Секция | Возможность |
| Группа | Поиск по номеру |
| Отображение списка всех групп |
| Отображение расписания |
| Добавление в избранные |
| Преподаватель | Поиск по фамилии имени и отчеству |
| Отображение списка всех преподавателей |
| Отображение расписания |
| Добавление в избранные |
| Расписание | Отображение по датам |
| Отображение по неделям |
| Выбор цвета для каждого типа занятия |
| Виджеты |
| Другое | Изменение иконки приложения |
| Очистка кэша |

Интерфейс данного приложения представлен на рисунке:

**

*а) б)*

Рисунок 1.4 – интерфейс приложения Bsuir Schedule:

*а* – расписание преподавателя; *б* – настройки

# Системное проектирование

Так как программное средство подразумевает большое количество функционала, в том числе и работу с базой данных и получение данных из API ИИС БГУИР, существует необходимость в создании предопределённой структуры.

Очевидно, что в первую очередь структуру можно представить как MVVM. Однако наиболее верным решением будет также дополнить её, кроме Модели, Представления и Модели Представления, добавить блоки работы с базой данных (далее – БД) и иные блоки, которые нельзя отнести к обычным Моделям.

Таким образом, структура проекта состоит из следующих блоков.

1. Модели;
2. Представления;
3. Модели Представлений;
4. Блок извлечения;
5. Блок декодирования;
6. Постоянное хранилище;
7. Фоновый контекст;
8. Контекст Представления.

Взаимосвязь между основными блоками проекта отражена на структурной схеме ГУИР.400201.01 С1.

## Модели

Модели будут являться наиболее крупной частью проекта, так как необходима не только реализация Моделей для всех данных из API ИИС, но и создание множества дополнительных Моделей для обработки логики. При этом использование

Swift предполагает широкое использование расширений, то есть описание Моделей в разных файлах и создание вместо новых Моделей расширений к существующим, часто это происходит в файле другой Модели. Например, для создания функции разделения массива занятий по датам их проведения принято не создавать глобальную функцию или метод для класса, в котором необходимо это действие, вместо этого создаётся расширение для массива элементом которого являются занятия. В таком случае функция будет вызываться от самого массива.

Написание расширений для Модели позволяет создавать наиболее универсальные функции и не заполнять глобальное пространство имён специфичными элементами.

Модели имеют связи с Моделями Представления, которые создают и изменяют первых, а блок декодирования может создавать новые и обновлять существующие Модели. Также Модели имеют связь с Фоновыми контекстами, где производятся фоновые вычисления.

## Представления

Структура SwiftUI подразумевает существование некого корневого Представления, из которого начинается работа с приложением. Из этого Представления существуют ссылки на другие представления, и так далее. При это всегда существует возможность вернуться назад по пути Представлений, также Представления путём ссылок могут приводить на самих себя.

В SwiftUI Представлением является всё, что отображается в интерфейсе, то есть, например, как кнопка является Представлением, которое описано программным декларативным образом, так и содержащий кнопку, например, список, также является Представлением, то есть одно Представление может включать в себя мужество других и даже самого себя. Благодаря этому свойству и возможности Представлений принимать переменные при инициализации, появляется возможность широкого использования шаблонов.

Представления должны иметь одинаковый функционал на всех платформах, для которых ведётся разработка, и быть реализованными наиболее универсальным методом, который поддерживает адаптивный графический пользовательский интерфейс.

Представление связано с Моделями Представлений: получает из них данные и отправляет данные на обработку. Также считается приемлемым и выполнять некоторые простые вычисления сразу в Представлениях, а в некоторых относительно простых или использующих большое шаблоны Представлениях вовсе не нужны Модели Представлений, они напрямую взаимодействуют с моделями. Это же касается и Представлений, которым не нужна какая-либо обработка данных, например, ячейка таблицы или кнопка.

Представление также взаимодействует с Контекстом Представления, из которого получает Модели и их обновлениях.

## Модели Представлений

Модели Представлений подразумеваются прослойкой между Моделями и Представлениями, в которых осуществляется специфичные для представления вычисления, неспецифичные же стоит выносить в сами Модели, а именно в расширения Моделей.

Модели Представлений чаще всего предназначены для обработки информации из Моделей для приведения их в советующий конкретному Представлению формат, примерном такой функциональности может служить вычисления периода длительности сессии для групп:, когда в Модели хранятся только даты начала и конца сессии, в Модели Представления из них вычитается длительность и уже этот высчитанный параметр передаётся для отображения пользователю в Представление.

Модель Представления имеет связь как с Моделью, так и с Представлением.

## Блок извлечения

Для получения необходимой информации из API ИИС БГУИР необходимо создание блока, который будет извлекать данные из API: отправлять запрос, получать и частично обрабатывать ответ.

Блок извлечения должен передавать полученные данные, либо информацию об ошибке, в блок декодирования.

## Блок декодирования

Блок декодирования получает данные из блока извлечения. Он должен быть реализован в соответствии структурами, представленными в API ИИС БГУИР, подробное их описание приведено в подразделе 1.1. Блок создаёт Модели в Фоновых контекстах и заполняет их декодированными данными. Для обновления Моделей должна быть реализована возможность получения существующих Моделей и записи в них новых извлечённых данных.

## Постоянное хранилище

Стандартным фреймворком для работы с БД CoreData является стандартным для iOS, iPadOS и macOS. Постоянное хранилище обеспечивает сохранение данных приложения в постоянную память устройства и их восстановление после выхода из приложения.

Постоянное хранилище является контейнером для хранения объектов CoreData, который позволяет сохранять, загружать и управлять данными приложения.

При использовании CoreData, данные приложения хранятся в базе данных SQLite, которая представляет собой один файл в контейнере приложения. Постоянное хранилище управляет этим файлом и предоставляет удобный интерфейс для взаимодействия с данными без использования SQL.

Данные из постоянного хранилища читаю и записывают различные контексты, которые могут автоматически обновляются при изменении данных в постоянном хранилище. При этом без выполнения операции сохранения контекста новые или изменённые данные не попадают не только в постоянное хранилище, но и в другие контексты.

Постоянное хранилище автоматически выполняет операции по объединению данных из сохранённого контекста постоянного хранилища в соответствии с заданными правилами. Например, если в постоянном хранилище имеется группа с определённым номером, и в одном из контекстов появилась группа с таким же номером, то, при сохранении контекста, хранилище может провести различные операции, которые зависят от настроек: полностью игнорировать данные из контекста; полностью игнорировать данные из хранилища; объединить объекты с приоритетом данных из хранилища; объединить объекты с приоритетом данных из контекста; выдать ошибку сохранения.

Постоянное хранилище связано с Контекстом Представления и Фоновыми контекстами.

## Фоновый контекст

Фоновые контексты используются для обработки данных, которые напрямую не отображаются пользователю, то есть они используются для выполнения задач на фоне, например, для загрузки данных из сети.

Основным отличим между фоновым контекстом и Контекстом Представления является то, что первый работает в отдельном потоке, в то время как второй работает в основном потоке, что означает, что задачи, выполняемые фоновым контекстом, не блокируют пользовательский интерфейс.

Фоновые контексты имеют связь как с Моделями, так и с блоком декодирования.

## Контекст Представления

В CoreData, главный контекст, он же Контекст Представления является контекстом, который обычно используется для работы с графическим интерфейсом в основном потоке приложения.

Контекст Представления представляет собой корневой объект контекстов CoreData, который создаётся автоматически при инициализации стека CoreData.

Основной задачей Контекста Представления является предоставление доступа к объектам из Постоянного хранилища Представлениям и другим основных функциям приложения, которые непосредственно связанных с Представлением. В отличие от других контекстов, Контекст Представления создаётся в главном потоке приложения и связан с главным потоком графического интерфейса, что позволяет ему выполняться в том же потоке, что и обработка графического интерфейса. Данный факт также накладывает и ограничения, Контекст Представления опасно использовать для загрузки и крупного изменения данных, так как при критической ошибке в нём велик шанс непредвиденного завершения всего приложения.

Контекст Представления обычно настроен таким образом, чтобы изменения, сделанные в нём, автоматически сохраняются в Постоянном хранилище, что гарантирует, что данные, которые отображают Представления, всегда будут соответствовать актуальным данным в Постоянном хранилище.

Контекст Представления связан только с Представлением и Постоянным хранилищем.

# Функциональное проектирование

Функциональное проектирование – аналог соответствующего раздела аппаратного дипломного проекта. Это основной раздел ПЗ, дающий ключ к пониманию функционирования разрабатываемой программы и исчерпывающую информацию о её структуре с точки зрения описания данных и обрабатывающих их подпрограмм (функций и процедур). Обработка данных, как известно, является основной целью работы любой программы. Поэтому здесь описываются заданные константы, пользовательские переменные, внутренние и внешние массивы и так далее. При определении необходимости в какой-либо обработке данных вводится соответствующая подпрограмма. При использовании объектно-ориентированного подхода могут описываться структура и взаимоотношения между классами со ссылками на чертёж диаграммы классов. Если при программировании (особенно в системах визуального программирования) используются стандартные библиотеки, то упор должен делаться на описание самостоятельно разработанных фрагментов кода. Стандартные функции лишь упоминаются. При использовании базы данных описывается структура таблиц и связи между ними со ссылками на чертёж модели данных. Здесь же раскрываются и потоки данных между подпрограммами, т.е. входные и выходные аргументы процедур и функций. Рекомендуется структурировать раздел в соответствии с блоками, выделенными на этапе системного проектирования. Данный раздел должен сопровождать чертёж диаграммы последовательности и содержать ссылки на него. Могут быть ссылки на чертёж схемы данных.

## Модели

## Представления

## Модели Представлений

## Блок извлечения

Блок извлечения реализован путём создания расширений для стандартного класса URLSession.

### FetchDataType

FetchDataType является перечислением, которое используется в проекте для определения URL-адресов API ИИС БГУИР для получения различных данных.

В этом перечислении определены различные случаи, каждый из которых представляет собой URL-адрес для конкретного типа данных: faculties – URL-адрес для получения списка факультетов; specialities – URL-адрес для получения списка специальностей; auditories – URL-адрес для получения списка аудиторий; departments – URL-адрес для получения списка кафедр; groups – URL-адрес для получения списка групп; group – URL-адрес для получения расписания для определенной группы; groupUpdateDate – URL-адрес для получения даты последнего обновления расписания для определенной группы; employees – URL-адрес для получения списка преподавателей; employee – URL-адрес для получения расписания для определенного преподавателя; employeeUpdateDate – URL-адрес для получения даты последнего обновления расписания для определенного преподавателя;

Перечисление использует тип String для хранения URL-адреса. В коде он используется для создания URL-объектов, которые используются для отправки запросов на сервер. Каждый случай перечисления имеет свою собственную строку, которая представляет URL-адрес для конкретного типа данных.

### Функции для извлечения данных их API ИИС БГУИР

Функция data(from:) async throws -> Dataиспользуется для получения данных с определенного URL-адреса. Она принимает один параметр: URL-адрес в виде строки.

Функция проверяет, может ли URL быть создан из заданной строки. Если URL не может быть создан, функция генерирует ошибку URLError(.badURL). Затем функция использует стандартный метод data(from:) для получения данных из сети. При успешном получении данных функция возвращает полученные данные в формате Data.

Функция data(for dataType: FetchDataType, with argument: String? = "") async throws -> Data? принимает тип запрашиваемых данных, который описан в enum FetchDataType, и необязательный аргумент (название группы или идентификатор преподавателя).

Функция использует эти параметры для создания URL-адреса, по которому будет выполнен запрос на сервер. Функция выполняет запрос на сервер с использованием метода await data(from url: URL) и возвращает данные в виде объекта Data. Если запрос произошёл успешно, то функция возвращает запрошенные данные. Если возникла ошибка (например, не удалось создать URL-адрес или сервер вернул пустой ответ), то функция генерирует ошибку типа URLError и возвращает nil.

Обе функции используют async/await для асинхронного получения данных из сети. Это означает, что при вызове этих функций процесс выполнения не блокируется, а продолжает выполнять другие задачи, пока данные не будут получены.

### Извлечение всех объектов определённого типа

Описанные выше функции являются универсальными и служат только для извлечения абстрактных данных из API. Извлечение и обработка конкретных данных производится в расширениях моделей. Ниже приведены необходимые функции для извлечения и обновления преподавателя из API.

Функция static func fetchAll() async используется для асинхронной загрузки данных о преподавателях из внешнего источника.

Функция начинается с загрузки данных с помощью URLSession.shared.data(from:), который был описан выше. Затем данные преобразуются из формата JSON в массив словарей ([[String: Any]]) с помощью JSONSerialization.jsonObject(with:options:).

Далее функция создает новый фоновый контекст Core Data с помощью newBackgroundContext() и задает политику слияния NSMergeByPropertyStoreTrumpMergePolicy, которая обеспечивает слияние объектов с приоритетом данных из контекста при сохранении в постоянное хранилище.

Затем используется JSONDecoder, который настроен на работу с Core Data. Информация о контексте (userInfo[.managedObjectContext]) и наличии вложенного контейнера групп (userInfo[.groupEmbeddedContainer]) используется для декодирования JSON-объектов в объекты модели CoreData.

Далее функция вызывает метод getAll(from:), который возвращает все объекты модели Employee из указанного контекста CoreData. Результат сохраняется в массив employees.

Затем функция итерируется по полученному массиву словарей и декодирует каждый словарь в соответствующий объект Employee. Если объект с таким же id уже существует в массиве employees, то происходит обновление объекта из данных существующего объекта с помощью JSONDecoder.update(\_:from:). Если же объекта с таким id нет, то происходит добавление нового объекта в массив employees.

В конце функция сохраняет контекст CoreData с помощью backgroundContext.perform(schedule: .immediate), чтобы сохранение происходило в фоне. В лог также выводится количество загруженных сотрудников и время выполнения функции.

### Обновление объекта

Функция обновление также приведена на примере обновления Employee. Функция func update() async -> Employee? является методом расширения для класса Employee и предназначена для обновления информации о преподавателе.

Она использует свойство Employee urlID, чтобы создать URL-адрес, по которому можно получить обновленные данные о преподавателе из API, используя URLSession.

Если данные были успешно загружены, они декодируются из JSON в формат Employee, используя JSONDecoder. Эта операция выполняется в фоновом контексте CoreData, который был создан с помощью newBackgroundContext() из PersistenceController, чтобы избежать блокировки пользовательского интерфейса.

Если данные были успешно декодированы, то объект Employee в фоновом контексте обновляется с помощью JSONDecoder.update(\_:from:).

Затем производится сохранение изменений в фоновом контексте с помощью backgroundContext.save().

В конце, функция возвращает обновленный объект Employee. Если данные не были загружены или произошла ошибка декодирования данных, то функция возвращает nil.

## Блок декодирования

Блок декодирования использует стандартные функции и протоколы JSONDecoder, Decodable и CodingKeys, для которых написаны расширения.

### Расширения стандартных библиотек

Расширения JSONDecoder и PropertyListDecoder расширяют протокол DecodingFormat. Эти расширения определяют метод decoder(for data: Data), который возвращает объект Decoder для соответствующего формата данных. Этот метод используется при декодировании объектов, чтобы создать необходимый декодер.

Далее определен протокол DecoderUpdatable, который имеет метод update(from decoder: Decoder), который обновляет значения из переданного декодера.

Протокол DecodingFormat, который определяет метод decoder(for data: Data), который создает и возвращает объект декодера для соответствующего формата данных, и метод decode<T: Decodable>(\_ type: T.Type, from data: Data), который декодирует объект типа T из переданных данных.

Два класса NestedSingleValueDecodingContainer и NestedDecoder определяют контейнеры декодирования для вложенных объектов в KeyedDecodingContainer. NestedSingleValueDecodingContainer – это контейнер, который декодирует единственное значение, а NestedDecoder - контейнер, который декодирует вложенные объекты.

Расширение KeyedDecodingContainer определяет метод update<T: DecoderUpdatable>(\_ value: inout T, forKey key: Key, userInfo: [CodingUserInfoKey : Any] = [:]), который использует NestedDecoder для обновления значений объекта T из ключа key в KeyedDecodingContainer.

### Декодирование определённого типа

Для декодирования определённого типа, например группы, занятия, используются инициализаторы классов, которые в качестве единственного параметра принимают тип Decoder. Ниже приведён пример такого инициализатора для группы.

Функция init() принимает параметр decoder типа Decoder и обновляет свойства объекта типа Group на основе полученных данных из Decoder.

Сначала функция извлекает контейнер с помощью метода container(keyedBy:) и передает туда тип CodingKeys Затем она проверяет, есть ли вложенный контейнер groupNestedContainer с помощью nestedContainer(keyedBy:,forKey:). Если контейнер есть, он присваивается переменной container, а если его нет, используется контейнер верхнего уровня.

Функция извлекает несколько свойств из контейнера с помощью метода decode(\_:forKey:), например, id, course, educationDegree, numberOfStudents, educationStart, educationEnd, examsStart, examsEnd. Если свойство numberOfStudents присутствует, оно также извлекается и присваивается свойству numberOfStudents объекта типа Group.

Затем функция проверяет наличие свойства specialityID в контейнере. Если оно присутствует, функция создает объект типа Speciality с помощью метода Speciality(from:) и присваивает его свойству speciality объекта типа Group, тем самым связав их в CoreData.

Затем функция извлекает свойства lessons и exams из контейнера и, если они присутствуют, присваивает объекту типа Group текущую дату как дату последнего обновления свойств lessonsUpdateDate.

В конце функция выводит сообщение в лог с информацией о том, что группа была обновлена и сколько времени это заняло.

## Постоянное хранилище

## Фоновый контекст

## Контекст Представления

# Разработка программных модулей

Разработка программных модулей̆ – аналог разработки принципиальной схемы аппаратного дипломного проекта. В этом разделе подробно описываются уже внутренние алгоритмы ключевых процедур и функций с разбиением на отдельные подразделы. Здесь же описывается реализация наиболее интересных алгоритмов, например, алгоритмов шифрования. Данный раздел может сопровождать чертежи схем программ и содержать ссылки на них.

# Программа и методика испытаний

В разделе, посвящённом программе и методике испытаний, описываются внутренние (если самотестирование заложено в программу) и внешние средства тестирования. Могут использоваться как оригинальные, так и стандартные тесты. Рассматриваются способы проверки надёжности (устойчивости, стабильности и т. д.) разработанной программы в различных режимах, включая многопользовательский и многозадачный режимы, а также корректность обработки входных, промежуточных и выходных данных, в том числе: в области граничных значений допустимых диапазонов, заведомого неправильных данных, файлов большого размера. Для каждого из тестов приводятся исходные данные, параметры и результаты.

# Руководство пользователя

В руководстве пользователя даётся описание работы с программой. Указываются требования к аппаратному и программному обеспечению. Описывается процесс инсталляции с указанием каталогов, ключей реестра конфигурационных файлов и так далее. Также описывается пользовательский интерфейс с указанием элементов управления (пунктов меню, кнопок, закладок и т. д.), режимов работы и последовательности действий. Здесь могут приводиться скриншоты работы программы.

# Технико-экономическое обоснование разработки и реализации на рынке ПРОГРАММНОГО средства для просмотра и обработки информации о расписании занятий

## Характеристика программного средства, разрабатываемого для реализации на рынке

Созданный дипломный проект представляет собой кроссплатформенное приложение для ОС iOS, iPadOS и macOS, которое позволяет студентам и сотрудникам БГУИР просматривать расписание занятий и экзаменов для групп, преподавателей и кабинетов, а также общее расписание для нескольких сущностей; просматривать подробную информацию о группах, преподавателях, кабинетах, специальностях; добавлять собственные занятия; добавлять и отслеживать задания.

Целью разработки данного проекта является упрощение получения и обработки данных о занятиях и другой информации. Приложение предназначено для использования в учебных целях и помогает студентам и сотрудникам эффективнее планировать своё время, управлять заданиями и получать своевременную информацию об изменениях в учебном расписании.

Целевой аудиторией данного приложения являются студенты, и сотрудники БГУИР, которым необходим удобный инструмент для получения информации из API ИИС. Также потенциальными покупателями могут быть иные учебные заведения, которые могут быть заинтересованы в использовании данного приложения.

На iOS в момент разработки существует два приложения со схожим функционалом: BSUIR Schedule, BSUIR Timetable. Существующие решения используют данные ИИС БГУИР не в полном объёме. Функционал этих решений ограничивается отображением расписания преподавателей и групп, однако API предоставляет больше информации: кабинеты и их типы, название специальностей, факультетов и кафедры, отношение преподавателей к кафедрам и другое. На iPadOS существует только BSUIR Schedule, а на macOS отсутствуют какие-либо приложения с таким функционалом.

Планируется распространение приложения через Apple App Store с использованием модели монетизации с бесплатной базовой версией и платной расширенной версией, которая имеет больший функционал.

## Расчёт инвестиций в разработку программного средства

### Расчёт зарплат на основную заработную плату разработчиков

Расчёт затрат на основную заработную плату разработчиков производится исходя из количества людей, которые занимаются разработкой программного продукта, месячной зарплаты каждого участника процесса разработки и сложности выполняемой ими работы. Затраты на основную заработную плату рассчитаны по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.1) |

где коэффициент премий и иных стимулирующих выплат;

категории исполнителей, занятых разработкой программного средства;

часовая заработная плата исполнителя *i*-й категории, р;

трудоемкость работ, выполняемых исполнителем *i*-йкатегории, ч.

Разработкой всего приложения занимается инженер-программист, Обязанности тестирования приложения лежат на инженере-тестировщике. Задачами инженера-программиста, который занимается являются создание модели данных, графического интерфейса, связи между моделью данных и графическим интерфейсом. Инженер-тестировщик занимается выявлением неработоспособных частей приложения, а также оценивает пользовательский опыт, получаемый от использования приложения.

Месячная заработная плата основана на медианных показателях для Junior инженера-программиста за 2023 год по Республике Беларусь, которая составляет 910 Долларов США в месяц, а для Junior инженера-тестировщика – 600 Долларов США [[[10]](#endnote-11)]. По состоянию на 15 апреля 2023 года, 1 Доллар США по курсу Национального Банка Республики Беларусь составляет 2,9441 Белорусских рублей [[[11]](#endnote-12)].

В перерасчёте на Белорусские рубли месячные оклады для инженера-программиста и инженера-тестировщика соответственно составляют составляет 2 679,13 и 1 760,46 Белорусских рублей соответственно.

Часовой оклад исполнителей высчитывается путём деления их месячного оклада на количество рабочих часов в месяце, то есть 160 часов.

За количество рабочих часов в месяце для инженера-программиста и инженера-тестировщика принято соответственно 196 и 32 часа.

Коэффициент премии приравнивается к единице, так как она входит сумму заработной платы. Затраты на основную заработную плату приведены в таблице:

Таблица 7.1 – Затраты на основную заработную плату

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Категория исполнителя | Месячный оклад, р | Часовой оклад, р | Трудоёмкость работ, ч | Итого, р |
| Инженер-программист | 2 679,13 | 16,74 | 196 | 3 281,04 |
| Инженер-тестировщик | 1 760,46 | 11,00 | 32 | 352,00 |
| Итого | | | | 3 633,04 |
| Премия и иные стимулирующие выплаты (0%) | | | | 0 |
| Всего затраты на основную заработную плату разработчиков | | | | 3 633,04 |

### Расчёт затрат на дополнительную заработную плату разработчиков

Расчёт затрат на дополнительную заработную плату команды разработчиков рассчитывается по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.2) |

где норматив дополнительной заработной платы.

Значение норматива дополнительной заработной платы принимает за 10 %.

### Расчёт отчислений на социальные нужды

Размер отчислений на социальные нужды определяется согласно ставке отчислений, которая на апрель 2023 г. равняется 35%: 29% отчисляется на пенсионное страхование, 6% − на социальное страхование. Расчёт отчислений на социальные нужды вычисляется по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.3) |

где норматив отчислений в ФСЗН.

### Расчёт прочих расходов

Расчёт затрат на прочие расходы определяется при помощи норматива прочих расчётов. Эта величина имеет значение 30%. Расчёт прочих расходов вычисляется по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.4) |

где норматив прочих расходов.

### Расчёт расходов на реализацию

Для того, чтобы рассчитать расходы на реализацию, необходимо знать норматив расходов на неё. Принимаем значение норматива равным 3%. Формула, которая использована для расчёта расходов на реализацию:

|  |  |  |
| --- | --- | --- |
|  |  | (7.5) |

где Нр – норматив расходов на реализацию.

### Расчёт общей суммы затрат на разработку и реализацию

Определяем общую сумму затрат на разработку и реализацию как сумму ранее вычисленных расходов: на основную заработную плату разработчиков, дополнительную заработную плату разработчиков, отчислений на социальные нужды, расходы на реализацию и прочие расходы. Значение определяется по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.6) |

Таким образом, величина затрат на разработку программного средства высчитывается по указанной выше формуле и указана в таблице:

Таблица 7.2 – Затраты на разработку

|  |  |  |
| --- | --- | --- |
| Название статьи затрат | Формула/таблица для расчёта | Значение, р. |
| 1. Основная заработная плата разработчиков | См. таблицу 7.1 | 3 633,04 |
| 2. Дополнительная заработная плата разработчиков |  | 363,30 |
| 3. Отчисление на социальные нужды |  | 1 398,71 |
| 4. Прочие расходы |  | 1 089,91 |
| 5. Расходы на реализацию |  | 108,99 |
| 6. Общая сумма затрат на разработку и реализацию |  | 6 593,95 |

## Расчёт экономического эффекта от реализации программного средства на рынке

Для расчёта экономического эффекта организации-разработчика программного средства, а именно чистой прибыли, необходимо знать такие параметры как объем продаж, цену реализации и затраты на разработку.

Соответственно необходимо создать обоснование возможного объёма продаж, количества проданных лицензий расширенной версии программного средства, купленного пользователями. В БГУИР обучается примерно 16 000 студентов, а работают в университете более 2 200 человек, что в сумме составляет примерно 18 000 человек [[[12]](#endnote-13)]. Процент пользователей в Республике Беларусь, которые используют iOS среди других мобильных ОС [[[13]](#endnote-14)], iPadOS среди других планшетных ОС [[[14]](#endnote-15)] и macOS среди других настольных ОС [[[15]](#endnote-16)] на март 2023 года соответственно составляют 36,97%, 40,08% и 14,13%.

Учитывая высокую для студентов стоимость устройств компании Apple, отсутствие информации о количестве пользователей, распределённом по возрастным и социальным группам, примем за процент перспективных пользователей приложения 30% от общего числа связанных с БГУИР людей, то есть 5 400 человек. Допустим, что из них 4 000 человек установят приложение, а из них 2 000 человек приобретут расширенную версию.

Стоимость в App Store задаётся по конкретным заранее определённым компанией уровням, то есть установить любую стоимость нельзя. Расчёты в белорусском регионе App Store ведутся в Долларах США. Так как функции расширенной версии не представлены ни в одном другом приложении и не имеют аналогов, наиболее оптимальной ценой данной версии предполагается 7,99 Долларов США, что с вычетом комиссии Apple в 30% составляет 5,6 Доллара США. Таким образом, отпускная цена копии программного средства составляет 16,48 Белорусских рубля.

Для расчёта прироста чистой прибыли необходимо учесть налог на добавленную стоимость, который высчитывается по следующей формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (7.7) |

где N – количество копий(лицензий) программного продукта, реализуемое за год, шт.;

– отпускная цена копии программного средства, р. ;

– количество приобретённых лицензий;

– ставка налога на добавленную стоимость, %.

Ставка налога на добавленную стоимость по состоянию на 15 апреля 2023 года, в соответствии с действующим законодательством Республики Беларусь, составляет 20%. Используя данное значение, посчитаем НДС:

|  |  |  |
| --- | --- | --- |
|  |  | (7.8) |

Посчитав налог на добавленную стоимость, можно рассчитать прирост чистой прибыли, которую получит разработчик от продажи программного продукта. Для этого используется формула:

|  |  |  |
| --- | --- | --- |
|  |  | (7.9) |

где *N* – количество копий(лицензий) программного продукта, реализуемое за год, шт.;

Цотп – отпускная цена копии программного средства, р.;

– сумма налога на добавленную стоимость, р.; Нп – ставка налога на прибыль, %;

– рентабельность продаж копий;

– рентабельность продаж копий.

Ставка налога на прибыль, согласно действующему законодательству, по состоянию на 14.04.2023 равна 20%. Рентабельность продаж копий взята в размере 40%. Зная ставку налога и рентабельность продаж копий (лицензий), рассчитывается прирост чистой прибыли для разработчика:

|  |  |  |
| --- | --- | --- |
|  |  | (7.10) |

## Расчёт показателей экономической эффективности разработки и реализации программного средства на рынке

Для того, чтобы оценить экономическую эффективность разработки и реализации программного средства на рынке, необходимо рассмотреть результат сравнения затрат на разработку данного программного продукта, а также полученный прирост чистой прибыли за год.

Сумма затрат на разработку меньше суммы годового экономического эффекта, поэтому можно сделать вывод, что такие инвестиции окупятся менее, чем за один год.

Таким образом, оценка экономической эффективности инвестиций производится при помощи расчёта рентабельности инвестиций (Return on Investment, ROI). Формула для расчёта ROI:

|  |  |  |
| --- | --- | --- |
|  |  | (7.11) |

где – прирост чистой прибыли, полученной от реализации программного средства на рынке информационных технологий, р.;

Зр – затраты на разработку и реализацию программного средства, р.

|  |  |  |
| --- | --- | --- |
|  |  | (7.12) |

## Вывод об экономической целесообразности реализации проектного решения

Проведённые расчёты технико-экономического обоснования позволяют сделать предварительный вывод о целесообразности разработки данного программного продукта. Общая сумма затрат на разработку и реализацию составила 6 593,95 Белорусских рублей, а отпускная цена была установлена на уровне 16,48 Белорусских рублей.

Прирост чистой прибыли за год, исходя из предполагаемого объёма продаж в размере 2000 расширенных версий в год, составляет 8 789,33 Белорусских рублей. Рентабельность инвестиций за год составляет 33,29%.

Это означает, что разработка данного программного продукта является целесообразной и реализация программного средства по установленной цене имеет смысл.

Однако, следует учитывать возможные риски, связанные с конкуренцией со стороны аналогов, что может привести к незамеченности продукта на рынке. Кроме того, высокая рентабельность связана с рисками, и расчётные результаты были получены при предполагаемом объёме продаж в 2000 копий в год.

Тем не менее, при поддержке проект может получить долгосрочное и успешное развитие, и количество проданных копий может превысить предполагаемое количество.

Масштабирование проекта на другие учебные заведения может способствовать его успешному развитию. В целом, инвестирование в предложенный проект также оправдано.

Заключение

Список использованных источников

[Введите текст] [Электронный ресурс]. – [Введите текст] – Режим доступа: [Введите текст]– Дата доступа: 19.04.2023

[Введите текст] [Электронный ресурс]. – [Введите текст] – Режим доступа: [Введите текст]– Дата доступа: 19.04.2023

[Введите текст] [Электронный ресурс]. – [Введите текст] – Режим доступа: [Введите текст]– Дата доступа: 19.04.2023

[Введите текст] [Электронный ресурс]. – [Введите текст] – Режим доступа: [Введите текст]– Дата доступа: 19.04.2023

[Введите текст] [Электронный ресурс]. – [Введите текст] – Режим доступа: [Введите текст]– Дата доступа: 19.04.2023

ПРИЛОЖЕНИЕ А

(обязательное)

**Спецификация**

ПРИЛОЖЕНИЕ Б

(обязательное)

**Ведомость документов**

1. [] Интегрированная информационная система «БГУИР: Университет» [Электронный ресурс]. – Документация – Режим доступа: https://iis.bsuir.by/api – Дата доступа: 18.04.2023 [↑](#endnote-ref-2)
2. [] Apple Developer [Электронный ресурс]. – Model-View-Controller – Режим доступа: <https://developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html> – Дата доступа: 17.04.2023 [↑](#endnote-ref-3)
3. [] Apple Developer [Электронный ресурс]. – UIKit – Режим доступа: <https://developer.apple.com/documentation/uikit> – Дата доступа: 17.04.2023 [↑](#endnote-ref-4)
4. [] Apple Developer [Электронный ресурс]. – AppKit – Режим доступа: <https://developer.apple.com/documentation/appkit> – Дата доступа: 17.04.2023 [↑](#endnote-ref-5)
5. [] Apple Developer [Электронный ресурс]. – SwiftUI – Режим доступа: <https://developer.apple.com/xcode/swiftui/> – Дата доступа: 17.04.2023 [↑](#endnote-ref-6)
6. [] Apple Developer [Электронный ресурс]. – Interfacing with UIKit – Режим доступа: <https://developer.apple.com/tutorials/swiftui/interfacing-with-uikit> – Дата доступа: 17.04.2023 [↑](#endnote-ref-7)
7. [] Apple App Store [Электронный ресурс]. – BSUIR Timetable – Режим доступа: <https://apps.apple.com/by/app/bsuir-timetable/id1443305662> – Дата доступа: 19.04.2023 [↑](#endnote-ref-8)
8. [] Apple App Store [Электронный ресурс]. – Bsuir Schedule – Режим доступа: <https://apps.apple.com/by/app/bsuir-schedule/id944151090> – Дата доступа: 19.04.2023 [↑](#endnote-ref-9)
9. [] GitHub [Электронный ресурс]. – [Введите текст] – Режим доступа: <https://github.com/asiliuk/BsuirScheduleApp> – Дата доступа: 19.04.2023 [↑](#endnote-ref-10)
10. [] Интернет-издание «Dev.by» [Электронный ресурс]. – Зарплата в ИТ – Режим доступа: <https://salaries.devby.io> – Дата доступа: 15.04.2023 [↑](#endnote-ref-11)
11. [] Национальный банк Республики Беларусь [Электронный ресурс]. – Официальные курсы белорусского рубля по отношению к иностранным валютам, устанавливаемые Национальным банком Республики Беларусь ежедневно, на 15.04.2023 – Режим доступа: <https://www.nbrb.by/statistics/rates/ratesdaily.asp> – Дата доступа: 15.04.2023 [↑](#endnote-ref-12)
12. [] Белорусский государственный университет информатики и радиоэлектроники [Электронный ресурс]. – БГУИР сегодня – Режим доступа: <https://www.bsuir.by/ru/bguir-segodnya> – Дата доступа: 15.04.2023 [↑](#endnote-ref-13)
13. [] Statcounter GlobalStats [Электронный ресурс]. – Mobile Operating System Market Share Belarus – Режим доступа: <https://gs.statcounter.com/os-market-share/mobile/belarus> – Дата доступа: 15.04.2023 [↑](#endnote-ref-14)
14. [] Statcounter GlobalStats [Электронный ресурс] Tablet Operating System Market Share Belarus – Режим доступа: <https://gs.statcounter.com/os-market-share/tablet/belarus> – Дата доступа: 15.04.2023 [↑](#endnote-ref-15)
15. [] Statcounter GlobalStats [Электронный ресурс]. – Desktop Operating System Market Share Belarus – Режим доступа: <https://gs.statcounter.com/os-market-share/desktop/belarus> – Дата доступа: 15.04.2023 [↑](#endnote-ref-16)