Итоговая аттестация по программе “IOS Разработка”

Уважаемые студенты, перед вами задание для итогового проекта, который вы должны выполнить после прохождения курсов “Основы языка Swift”, “Разработка приложений на основе языка Swift” и “Objective-C для IOS разработчиков”.

Описание итоговой аттестации:

1) Цель задания: *повторить и отработать на практике материал, изученный в ходе программы.*

2) Проект включает в себя **обязательное задание** и **дополнительные задания**.

   - Обязательное задание необходимо выполнить для получения диплома. За него ставится оценка.

   - Дополнительные задание можно выполнить для портфолио и также сдать и получить обратную связь. Его выполнение полезно, но не влияет на оценку.

3) Инструменты, которые обязательно нужно использовать для **обязательного задания:** *Swift, Objective-C*

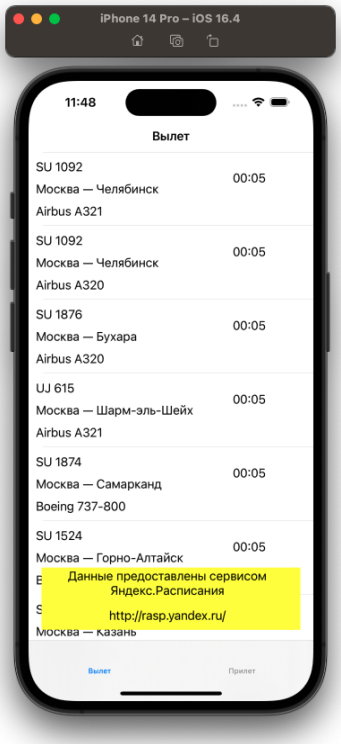
4) Формат сдачи: *ссылка на подписанную и доступную для просмотра копию данного шаблона с выполненными заданиями в синих полях. Инструкция.*

Обязательное задание (обязательно к выполнению):

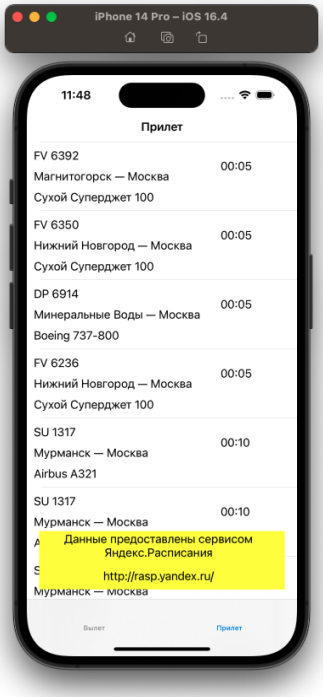
Задание 1.

Необходимо: разработать приложение, отображающее информацию о рейсах аэропорта.

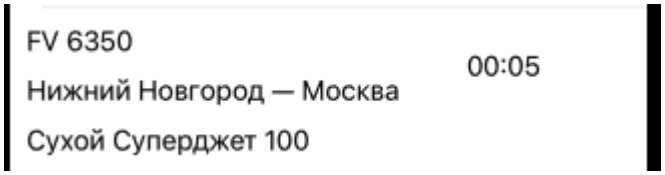
1. Для разработки используйте Swift
2. Для разработки проекта используйте UIKit. Удалите упоминания Main.storyboard
3. В проекте должно быть несколько экранов:
   a. На первом экране внизу есть вкладки, которые позволяют переключиться между табло вылета и прилета
   b. Экран вылета представляет собой таблицу со списком рейсов



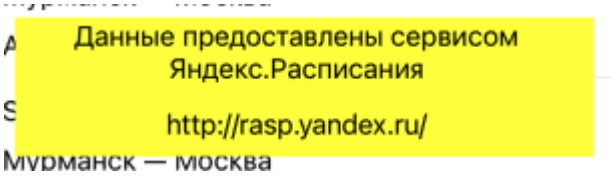
c. Экран прилета представляет собой таблицу со списком рейсов



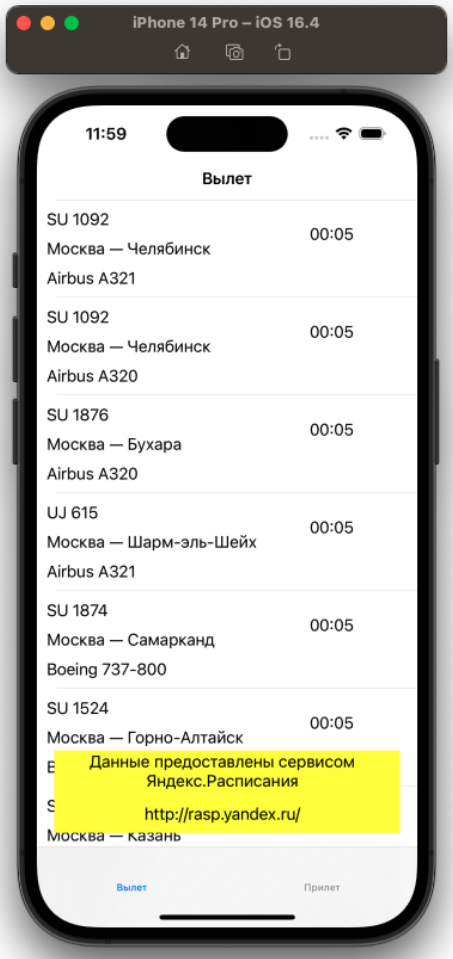
d. Ячейка должна содержать номер рейса, время вылета/прилета, название рейса, транспортное средство. Ячейка должна выглядеть следующим образом:

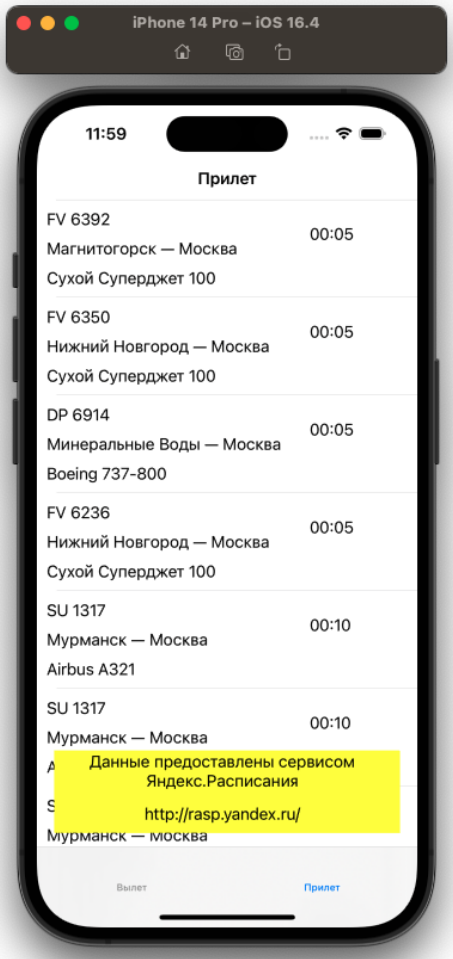


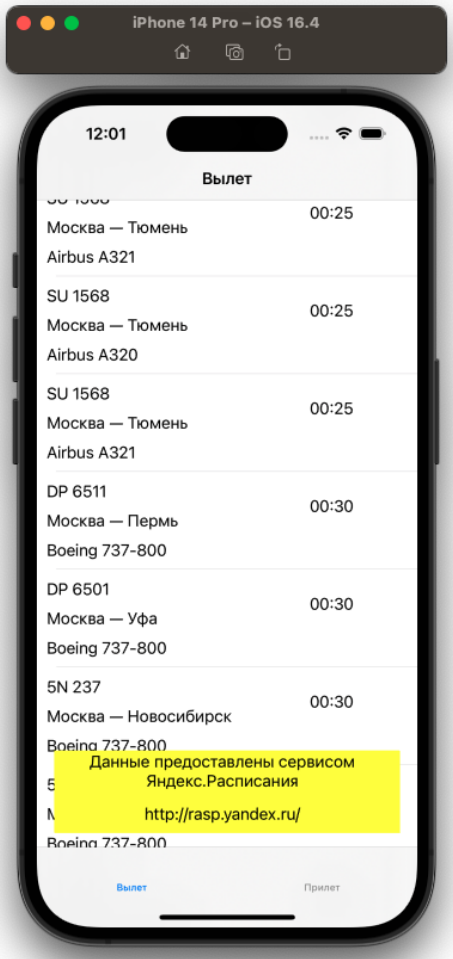
e. Не создавайте ячейки напрямую. Переиспользуйте ячейки.
f. Для запросов изучите: https://yandex.ru/dev/rasp/raspapi/. Вам будет необходимо получить расписание любого аэропорта на выбор. Дату добавлять не нужно, отобразите все рейсы, которые придут. Документация к методам здесь: https://yandex.ru/dev/rasp/doc/reference/schedule-point-point.html. Вам необходимо расписание рейсов по станции: https://yandex.ru/dev/rasp/doc/reference/schedule-on-station.html. Обратите внимание, что табло вылетов и прилетов должно быть для одного и того же аэропорта.
g. Внизу экранов должен располагаться баннер с информацией, кем предоставлен сервис. Запрос для получения информации можно посмотреть здесь: https://yandex.ru/dev/rasp/doc/reference/query-copyright.html. Отобразите текст и ссылку. Баннер должен быть виден всегда, то есть даже когда пользователь скроллит таблицу - баннер все равно остается на экране.

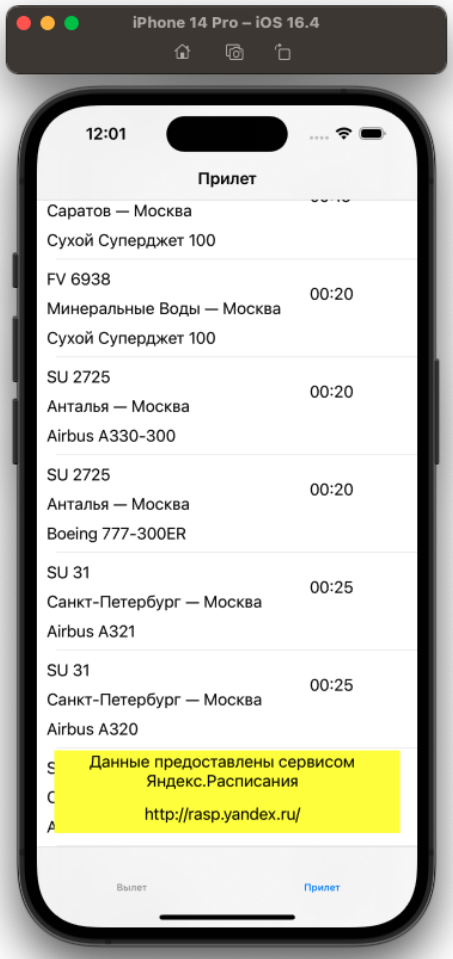


Результат, который у вас должен получиться:

Экран вылета	

Экран прилета	

Экран вылета после скролла таблицы. Баннер не должен исчезать с экрана.	

Экран прилета после скролла таблицы. Баннер не должен исчезать с экрана.	

Задание 2:

Для выполнения данного задания необходимо использовать Objective-C.

1. Создайте проект. Перейдите в File > New > Project, затем выберите “macOS” > “Command Line Tool”.
2. В main напишите программу. Необходимо решать квадратное уравнение типа ax^2+bx+c=0. Коэффициенты вводит пользователь. В результате выполнения должны выводиться корни уравнения. Например, если уравнение: x^2-2x-3=0, то вывод следующий:

```
Коэффициент a: 1
Коэффициент b: -2
Коэффициент c: -3
Первый корень: 3.000000, второй корень: -1.000000
```

   Если это уравнение типа 5x^2=0

```
Коэффициент a: 5
Коэффициент b: 0
Коэффициент c: 0
Один корень: 0.000000
```

Результат, который у вас должен получиться:

```
Коэффициент a: 1
Коэффициент b: 0
Коэффициент c: -25
Первый корень: 5.000000, второй корень: -5.000000
```

```
Коэффициент a: 3
Коэффициент b: 1
Коэффициент c: 2
Нет корней
```

```
Коэффициент a: 1
Коэффициент b: -5
Коэффициент c: 6
Первый корень: 3.000000, второй корень: 2.000000
```

```
Коэффициент a: 1
Коэффициент b: -6
Коэффициент c: 9
Один корень: 3.000000
```

Решение

В синие поля ниже вставьте ссылку на репозиторий, в котором лежит готовый проект или ссылку на гугл диск с проектом(разработанным приложением).

Обратите внимание, что нужно сдавать проект целиком, то есть не только .xcodeproj файл, но и все файлы, которые созданы в процессе.

Ссылка с проектом к заданию 1

Ссылка с проектом к заданию 2

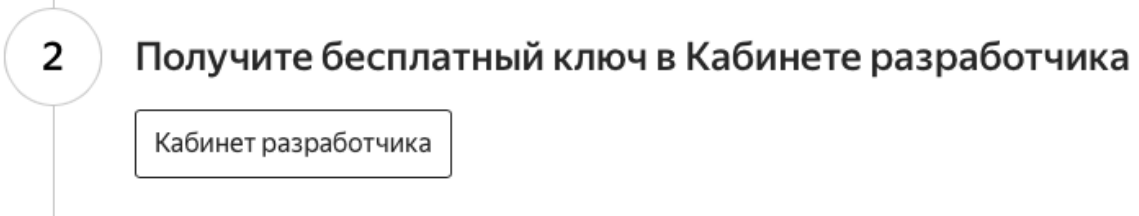
Дополнительное задание к заданию 1.

1. Выберите архитектурный паттерн. Разработайте приложения, используя этот архитектурный паттерн.
2. По клику на рейс необходимо переходить в карточку рейса. На экране должна отображаться та же информация, что в ячейке.
3. Добавить для пользователя возможность обновить таблицу. Дать понять пользователю, что таблица в процессе обновления.
4. Отобразите рейсы только на текущую дату. Не показывайте прошедшие рейсы.
5. Если приходит какая-то ошибка - предупредите пользователя, что не можете показать данные.
6. Сделайте баннер кликабельным и добавьте переход по отображаемой ссылке.
7. Покройте unit-тестами все приложение.

Блоки с подсказками:

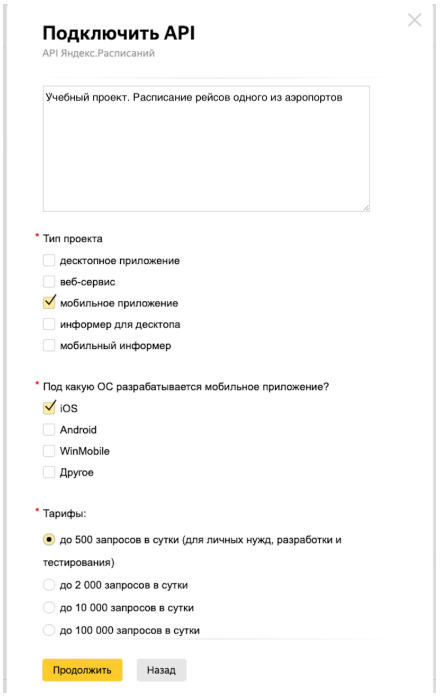
Подсказки для 1 задания:

1. Для того, чтобы получать информацию о рейсах будет необходим аккаунт в Яндекс. Если его нет - создайте.
2. После создания аккаунта создайте ключ, сделать это можно здесь: https://yandex.ru/dev/rasp/raspapi/, кликнув по “Кабинет разработчика”.



Скорее всего ключ активируется сразу. Рядом с ключом будет написано “Активен”.

3. При получении ключа выбираем API Яндекс.Расписаний. Заполнить поля можно следующим образом:



4. Для получения списка рейсов необходимо изучить документацию: https://yandex.ru/dev/rasp/doc/reference/schedule-on-station.html. Если не получается собрать запрос, он должен выглядеть следующим образом:
Запрос на вылетающие рейсы
"https://api.rasp.yandex.net/v3.0/schedule/?apikey=" + apiKey +
"&station=s9600213&transport_types=plane&event=departure"

Запрос на прилетающие рейсы
"https://api.rasp.yandex.net/v3.0/schedule/?apikey=" + apiKey +
"&station=s9600213&transport_types=plane&event=arrival"

В данных запросах apiKey - ключ, который был получен в предыдущих пунктах, то есть ваш API-ключ.
Также после station, вместо s9600213 может быть другой код станции, например:

s9628674	Бермуды
s9600216	Домодедово
s9600213	Шереметьево

s9600215	Внуково
s9600366	Пулково

4. Для отображения вам необходимы будут следующие поля: departure, arrival, number, title, vehicle. Обратите внимание, что они на разных уровнях. Создайте итоговую модель правильно.
5. Если не получается создать модель, обратите внимание на следующую схему:
   Структура1 {
   let schedule: [Структура2]
   }
   Структура2 {
   let thread: Структура3
   время вылета
   время прилета
   }
   Структура3 {
   номер рейса
   название рейса
   транспортное средство
   }
6. Для получения данных для баннера необходимо изучить следующую документацию: https://yandex.ru/dev/rasp/doc/reference/query-copyright.html. Вам необходимо отобразить текст и ссылку. Если не получается собрать правильный url, он должен быть следующий: "https://api.rasp.yandex.net/v3.0/copyright/?apikey=" + apiKey + "&format=json".
   В данном запросе apiKey - ключ, который был получен в предыдущих пунктах, то есть ваш API-ключ.
7. Если не получается создать модель, рассмотрите следующую схему:
   Структура1 {
   var copyright: Структура2
   }
   Структура2 {
   текст
   url
   }
8. Для баннера можно создать свой собственный кастомный элемент, на котором расположены два лейбла. Затем, на экране с таблицей добавить этот готовый элемент, вместо двух отдельных лейблов. Свой кастомный элемент должен быть наследником UIView.

9. Чтобы было можно переиспользовать ячейки не создавайте их напрямую, а используйте tableView.dequeueReusableCell. Не забывайте вызвать tableView.register заранее.

Подсказки для 2 задания

1. Коэффициенты должны вводиться пользователем. Для этого можете использовать scanf
2. Не забудьте, что в случае с квадратным уравнением есть три варианта:
    a. Дискриминант меньше нуля. Нет корней.
    b. Дискриминант равен нулю. Один корень,
    c. Дискриминант больше нуля. Два корня.

Ссылки:

1. Пример работы приложения