Пояснительная записка к контрольному домашнему заданию по дисциплине «Программирование»

Образовательная программа «Бизнес-Информатика»

Бакалавриат 1-й курс

Королев Кирилл ББИ168

Тема проекта

Симуляция работы банковского мобильного приложения

Адрес репозитория

<https://github.com/kirill-korolev/bankapp>

Описание функционала программы

Данный проект является прототипом реально существующих банковских приложений, таким образом, его целью было не полная реализация функционала, а лишь имплементация основных features. Некоторые из них были упрощены: например, не использовались инструменты криптографии для хранения и передачи данных, многие элементы интерфейса были нивелированы по причине ненадобности.

Говоря конкретнее о функционале, отмечу наличие двух-ступенчатой авторизации пользователя, то есть помимо основного пароля, пользователь должен ввести четырехзначный пин-код. Вся информация об авторизации хранится в так называемом хранилище устройства, и при повторном входе в приложение требуется ввести лишь пин-код.

После аутентификации пользователь попадает на экран «Главная», на котором приведена информация о его действительных банковских картах, интерфейс для открытия депозита и соответственно список открытых вкладов.

Пользователь имеет возможность подробнее ознакомиться с информацией о карте: может узнать не только о текущем балансе, номере карты и производителе, но и о ее типе (дебетовая, кредитная или овердрафтная), cvv-коде и ознакомиться с историей последних операций. Операции делятся на два типа: переводы физическим лицам и оплата услуг различных сервисов. В истории операций пользователь может ознакомиться, кому конкретно он осуществил перевод, узнать сумму перевода и дату.

У пользователя есть возможность открыть новый вклад, выбрав стартовую сумму и срок. После создания список текущих вкладов обновится и пользователь сможет проводить мониторинг своих данных.

Во вкладке «Переводы» у пользователя есть альтернативы: осуществить перевод между своими картами, либо перевести денежную сумму на счет другого клиента банка, введя его номер карты и сумму перевода.

Также пользователь может ознакомиться с полной историей своих операций, которые представлены в виде таблицы, хранящей информацию о номере операции, ее типе, сумме и дате проведения.

При соответствующем желании пользователь имеет возможность изменить свой пароль в личном кабинете: подтвердить свой старый пароль и ввести новый.

И наконец, пользователь может выйти из своего аккаунта и ввести иные данные для авторизации.

Используемые технологии и библиотеки

Данное iOS-приложение реализовано преимущественно на языке Swift, некоторые надстройки и библиотеки написаны на Objective-C, а своеобразная имитация RESTful API - на PHP.

В процессе разработки были использованы следующие библиотеки: Reachability (для проверки соединения с интернетом), CustomActivityIndicatorView (для создания кастомных элементов загрузки), SWRevealViewController (для создания бокового бургер-меню), JSSAlertView (для удобного создания алертов) и наконец SmileLock (интерфейс для четырехзначного пин-кода).

Приложение соответствует некоторому слиянию таких шаблонов проектирования, как: MVC (Model-View-Controller) и VIPER (Viewer-Interactor-Presenter-Entity-Router), в частности, в зависимости от сложности реализации конкретного экрана, большинство подзадач делегировались соответствующим классам, в целях минимизации объема кода в рамках одного класса и читаемости.

Приложение соответствует парадигме CRUD, предоставляя возможность создавать, читать, изменять и удалять данные.

Реализована защита от некорректного ввода: перед отправкой любой формы приложение выдает предупреждение о неправильности введенных данных, в частности при аутентификации, смене пароля и осуществлении переводов.

Соблюден единый стиль наименования и оформления кода, стиль написания слов соответствует CamelCase.

Приложение взаимодействует с несколькими сущностями, используя различные шаблоны проектирования, например: Facade, Singletone, Adapter.

В приложении реализована сериализация и десериализация данных в формате JSON при отправке и получении HTTP-запросов.

Навигация по приложению организована в виде бокового бургер-меню, позволяющее перемещаться по страницам приложения.

Функционал приложения подразумевает наличие двухступенчатой аутентификации пользователя. В свою очередь, так как приложение банковское, отсуствует возможность чтения без авторизации.

Все данные о пользователях, сервисах, транзакциях, вкладах и картах хранятся в базе данных MySQL на собственном сервере. Передача данных организована посредством HTTP-запросов (GET) на сервер, к примеру, такого вида: thehomeland.ru/service/deposit.php?sum=<sum>&term=<term>&owner=<owner>

Ответ сервера представляет из себя ассоциативный массив, а конкретнее массив данных в формате JSON.

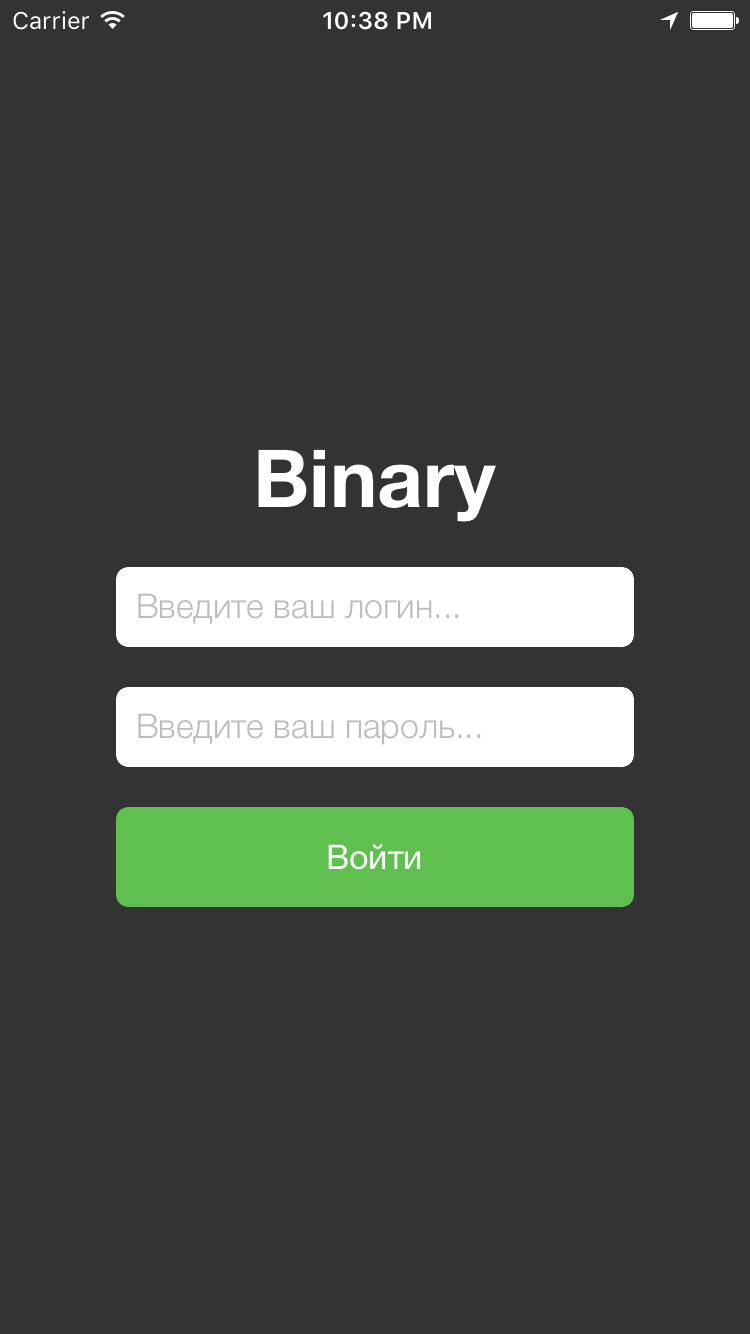
При отправке запросов модуль, отвечающий за загрузку данных, логирует результаты в консоль при успешном или неудачном результате отправки.

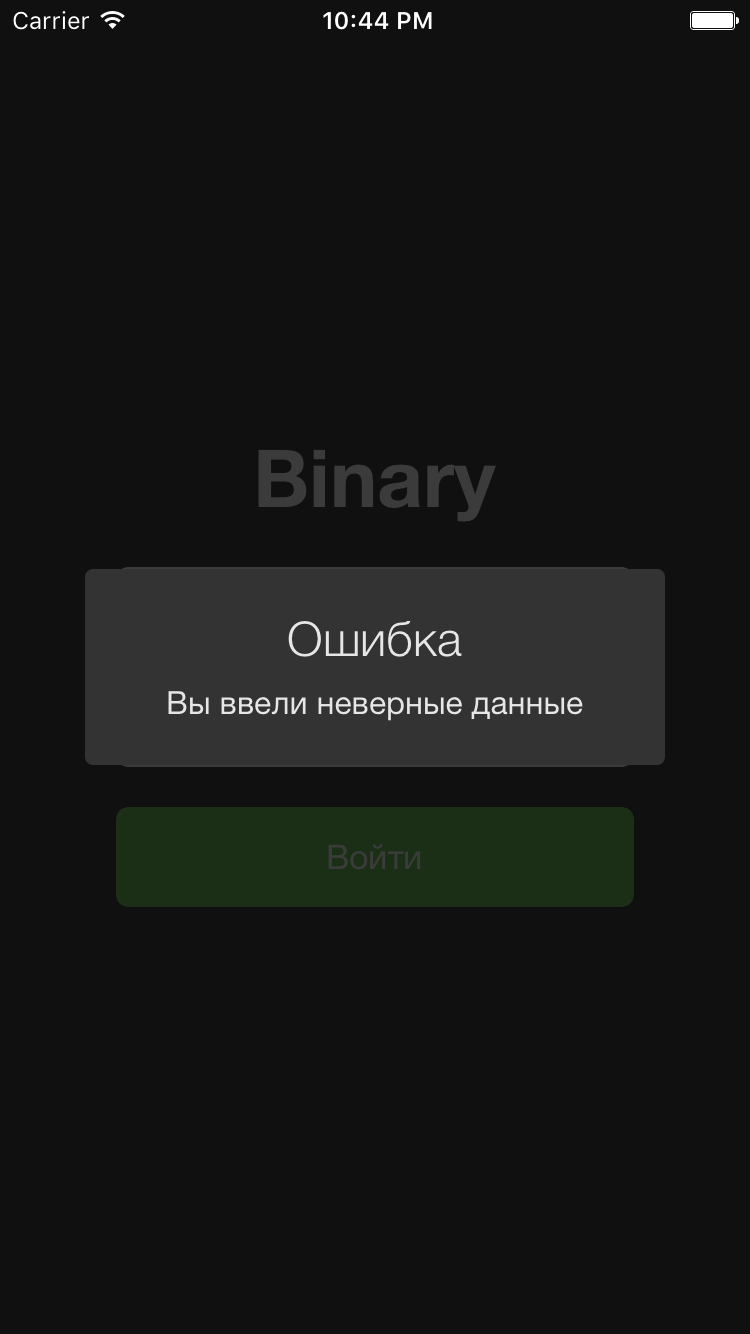
История изменений отражена в истории последних транзакций пользователя, как по каждой карте, так и в целом.

Приложение использует стандартные библиотеки Cocoa Touch – UIKit для реализации анимаций в приложении, в частности, при переходе на новый экран, отправки формы, появлении модального окна, отображении алертов.

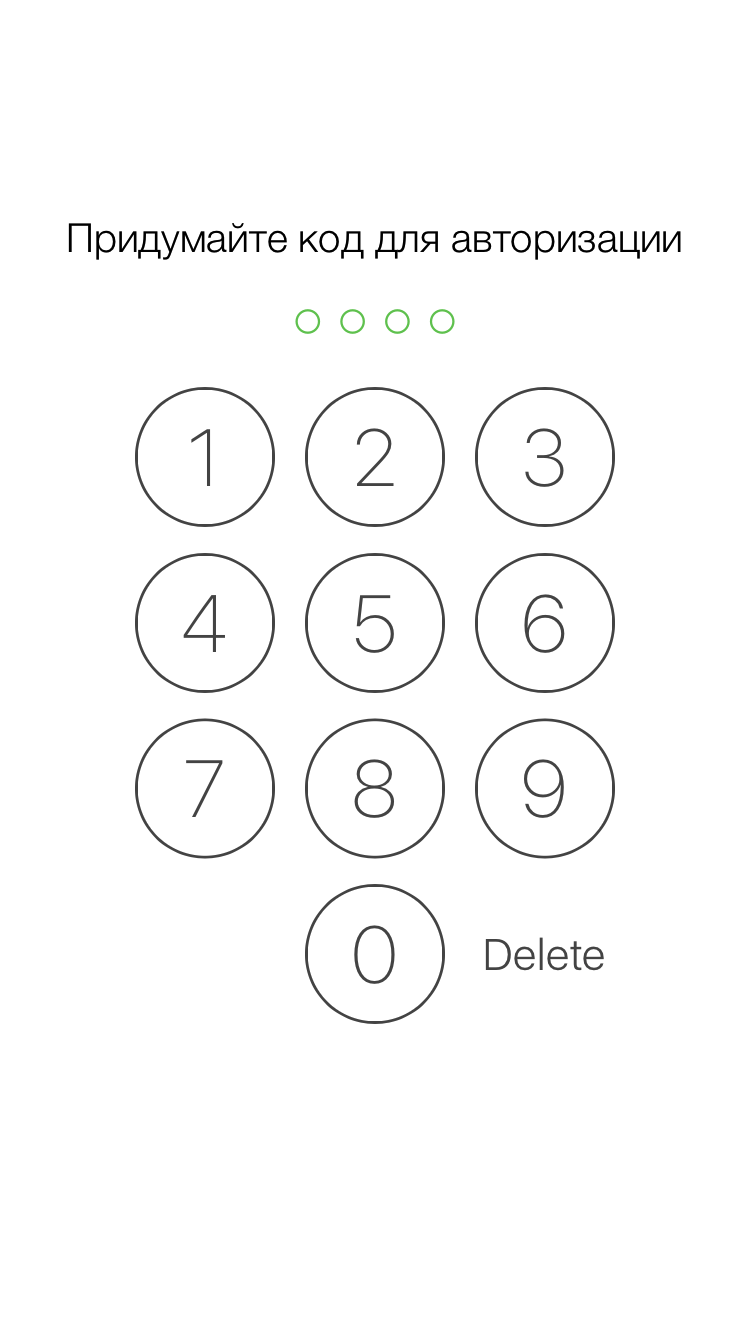
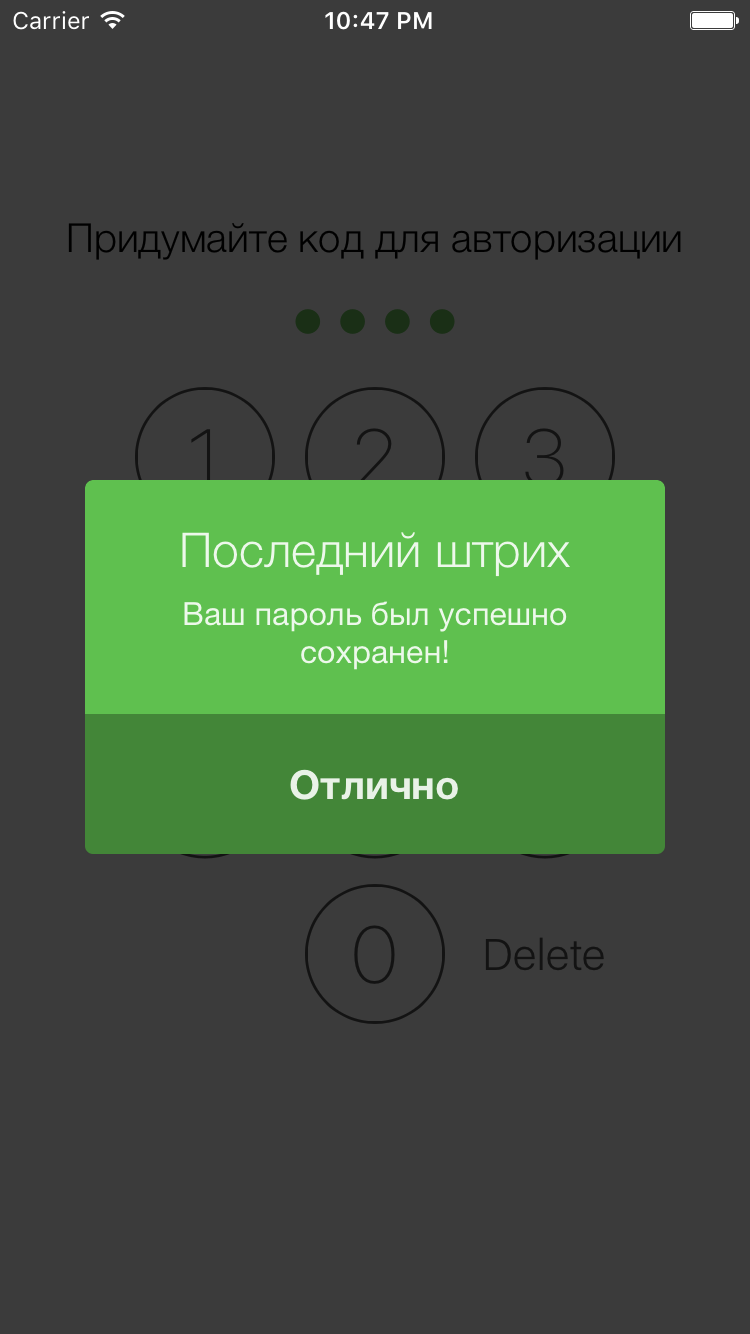
Интерфейс программы

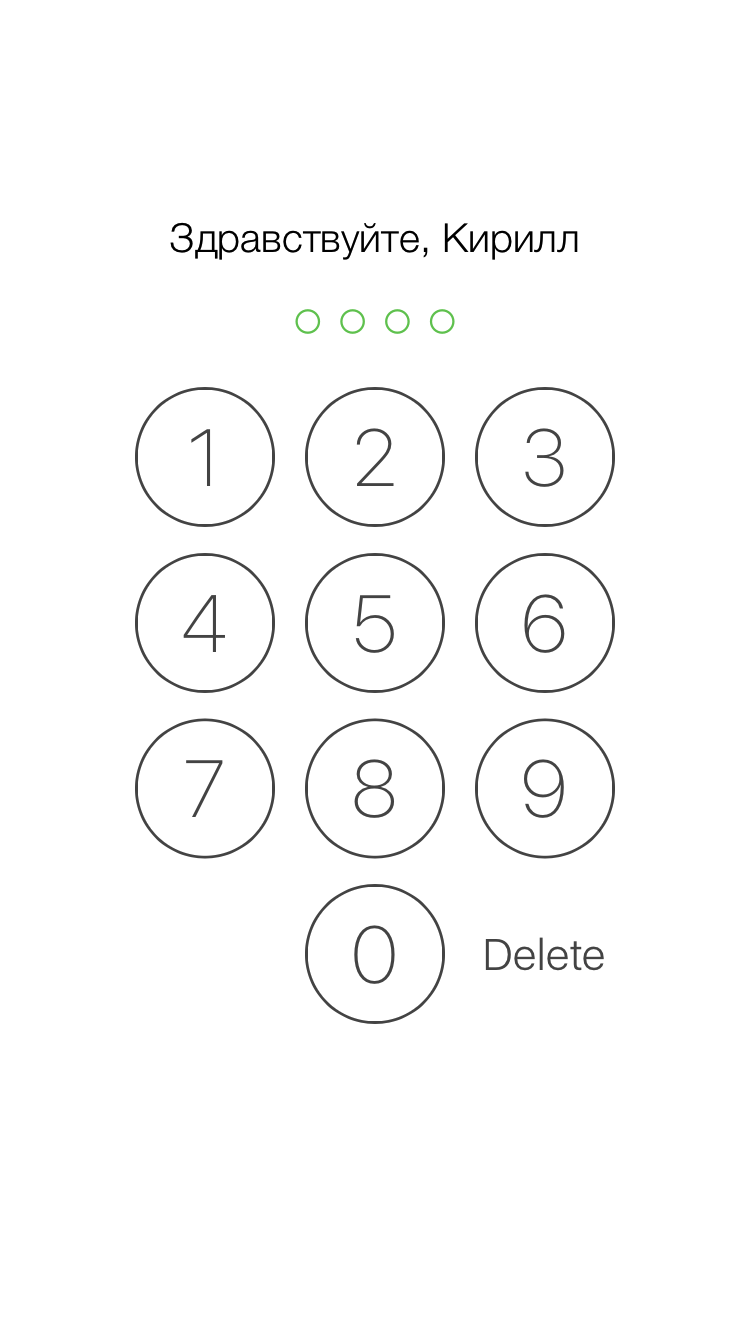
Аутентификация



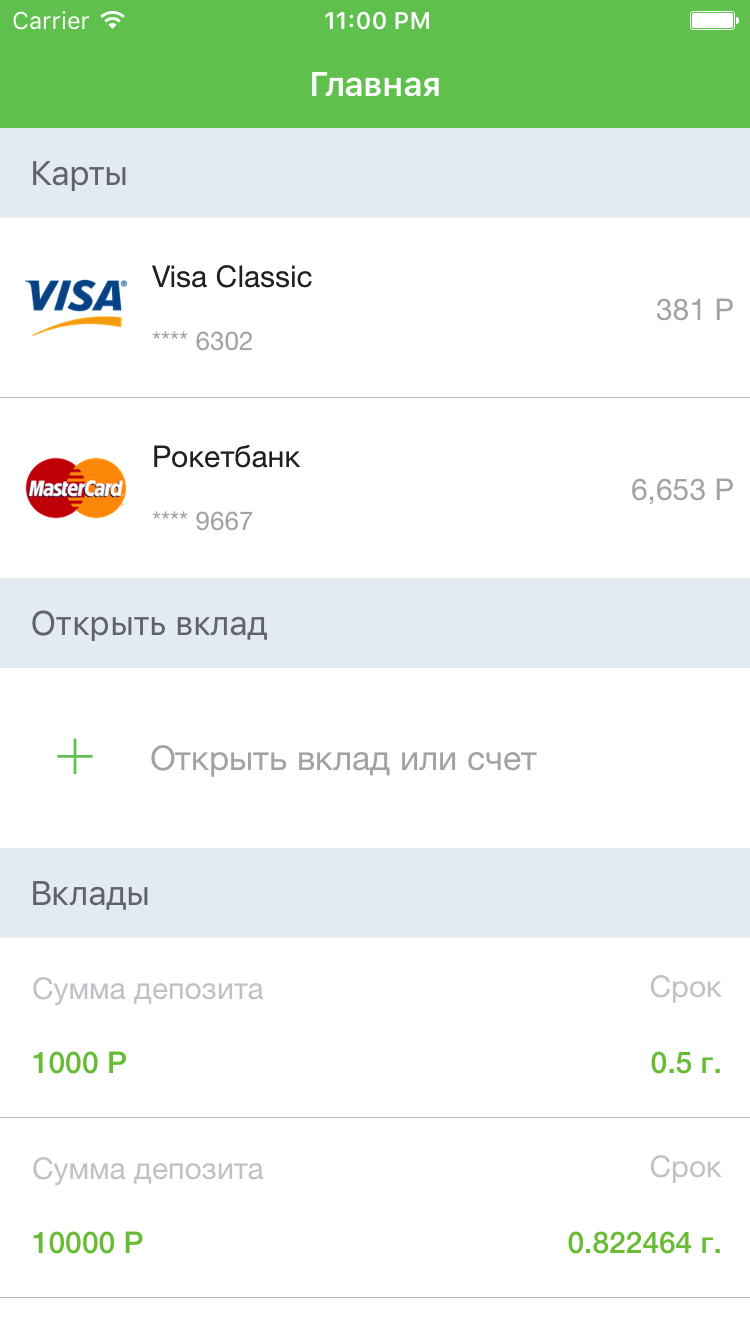
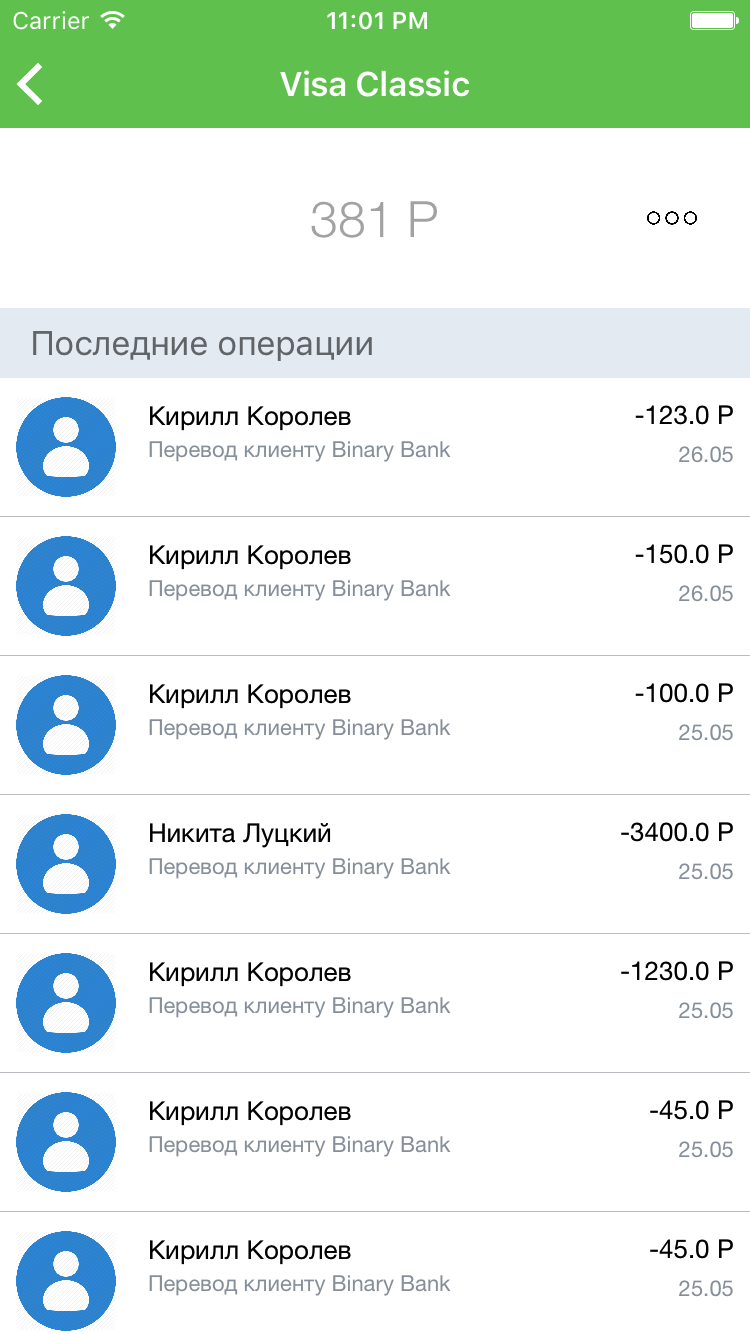
Для авторизации пользователь вводит свой логин и пароль. При вводе невалидных данных выводится соответствующее сообщение.

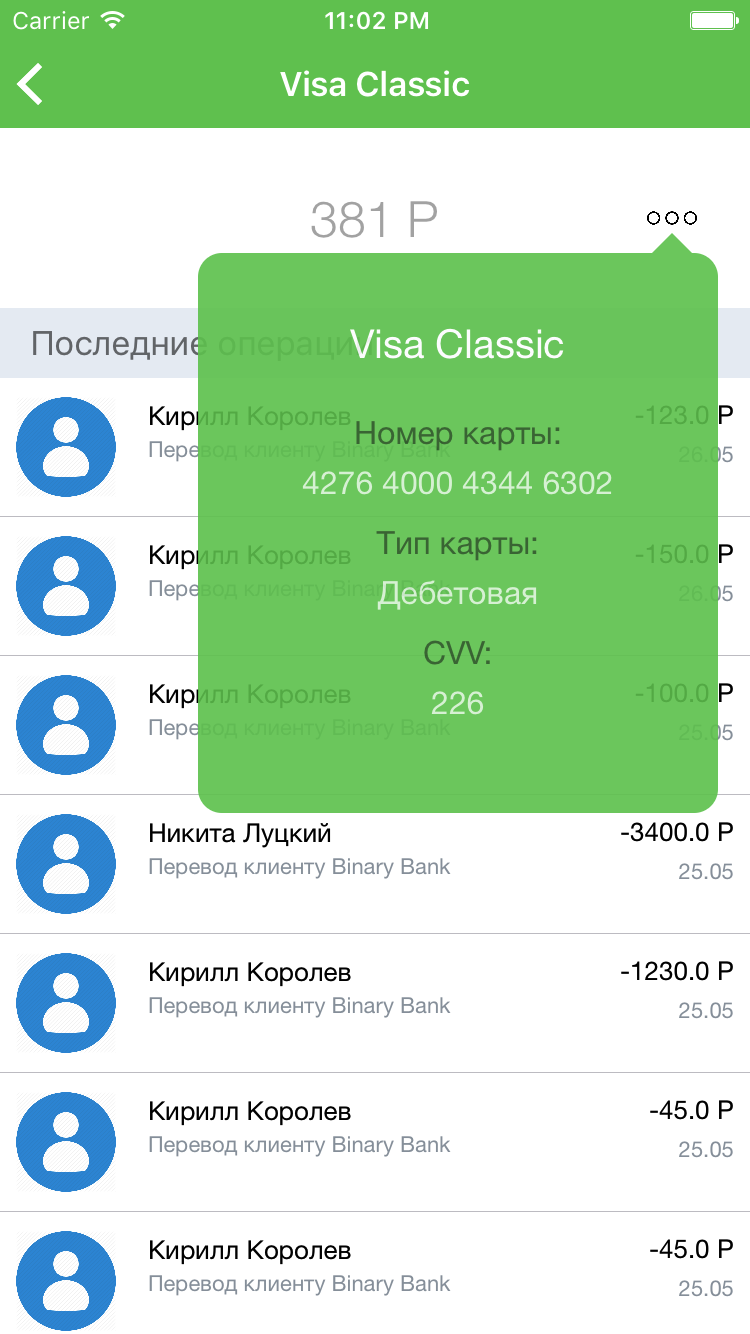
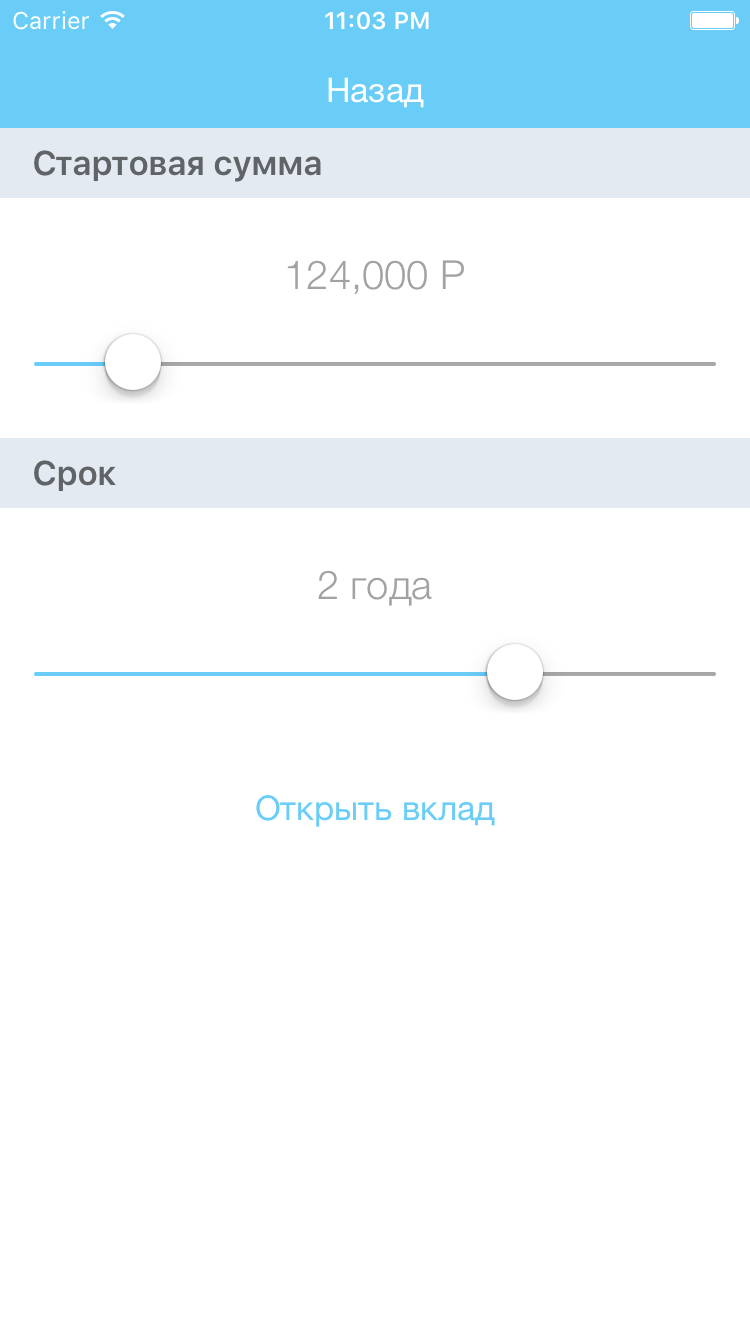
После успешного прохождения первого этапа авторизации пользователю предлагается придумать личный код, который он будет вводить при следующем входе в приложение.





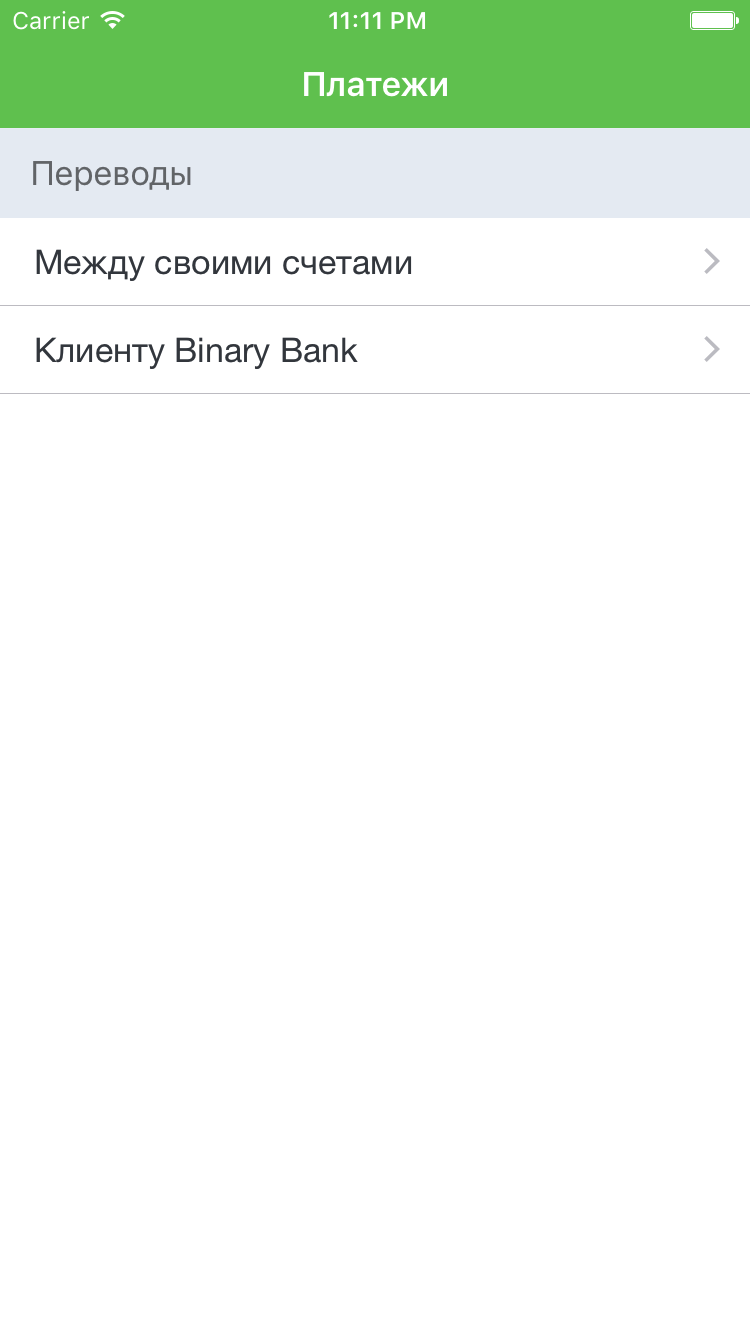
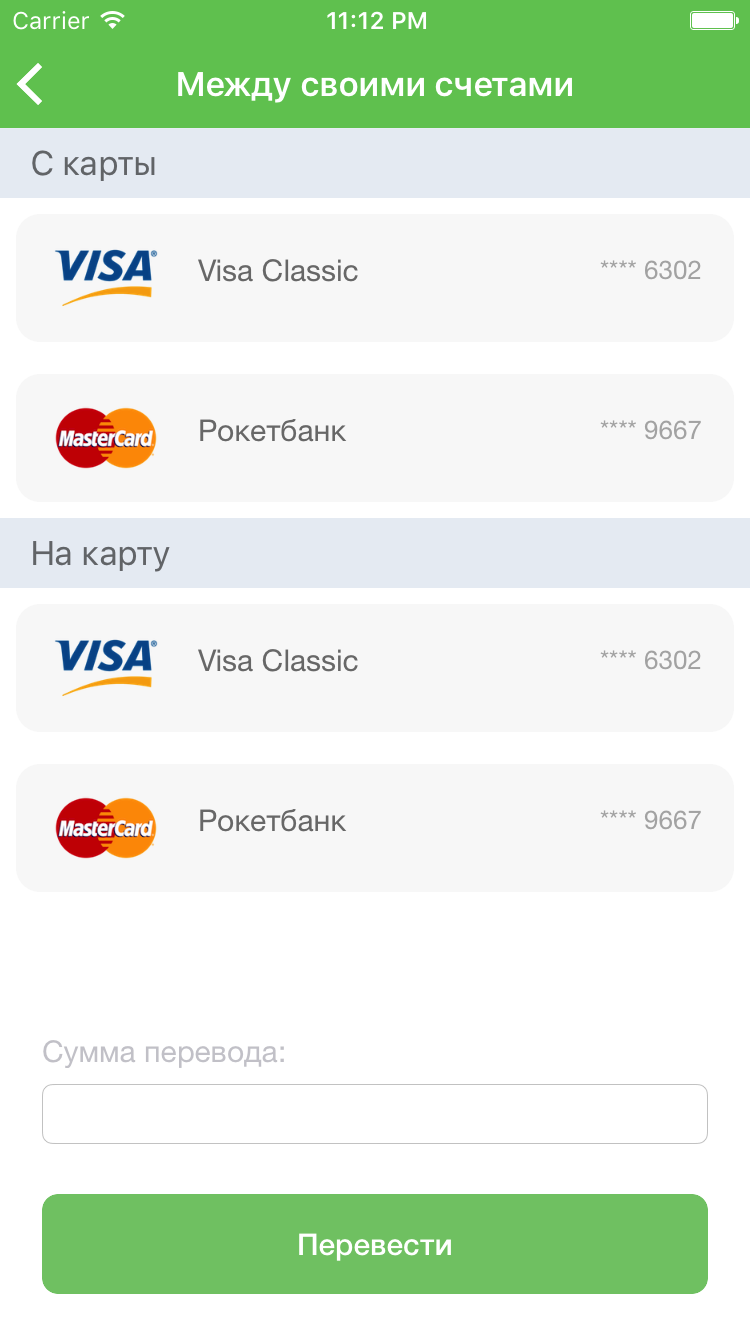
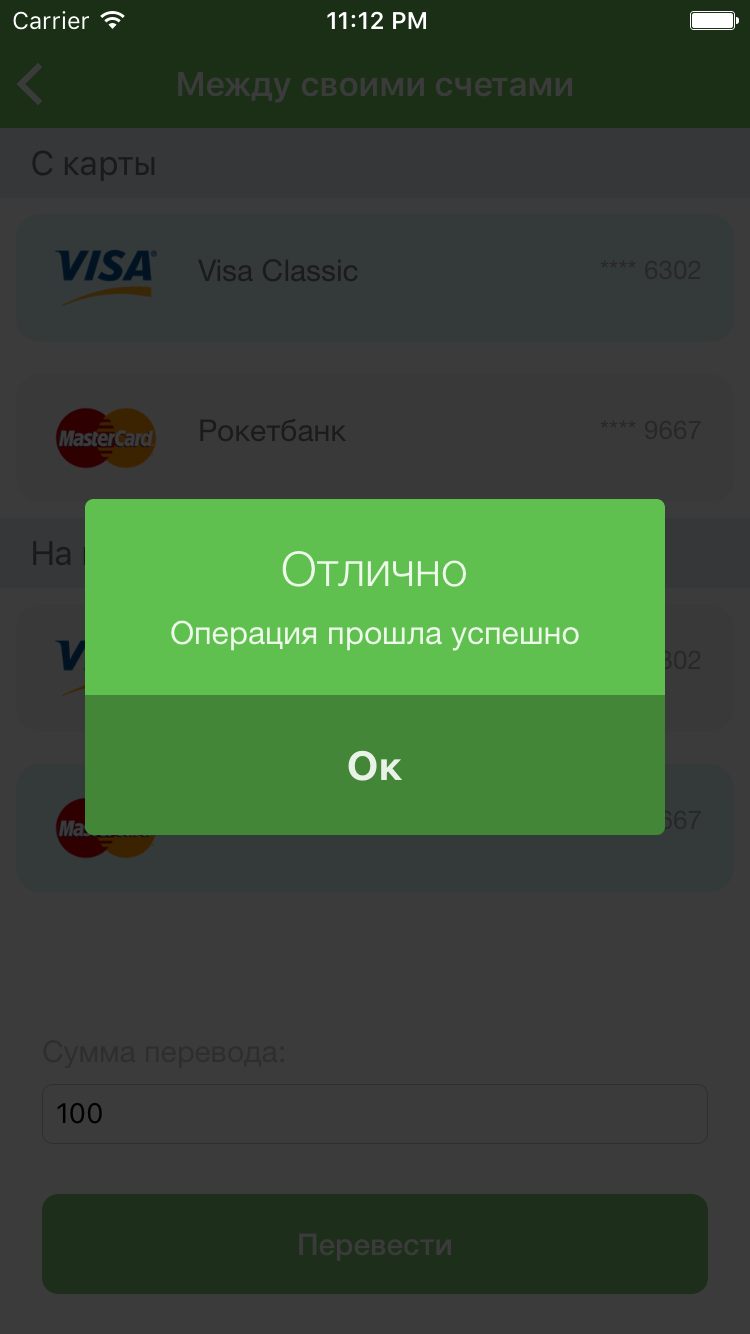
Главная страница

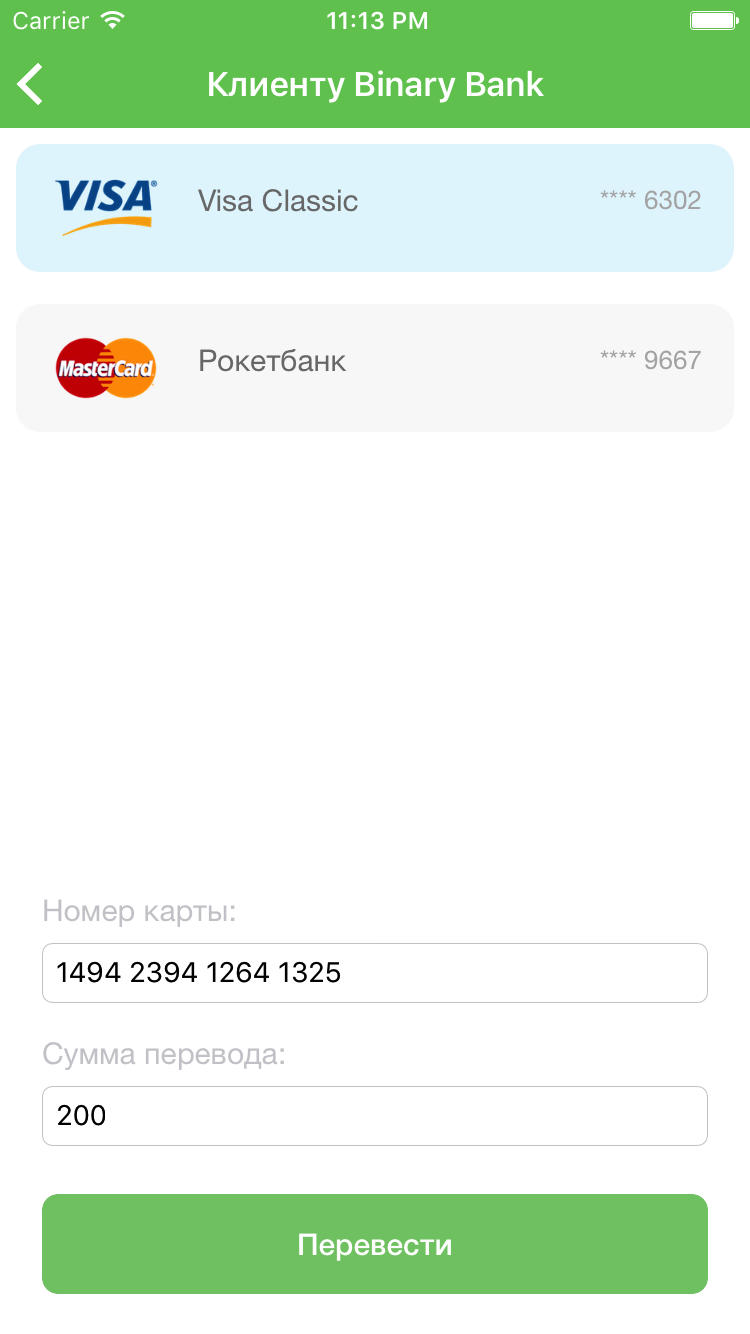


На главной странице приложения у пользователя есть три опции: посмотреть подробную информацию о карте, открыть новый вклад, либо проверить уже открытые вклады. Нажимая на карту, пользователь переходит на новый экран, на котором продемонстрирована основная информация о карте и история операций.

Нажав на кнопку “Открыть вклад или счет”, пользователь переходит на новый экран, на котором ему предлагается выбрать соответствующие условия и открыть вклад.

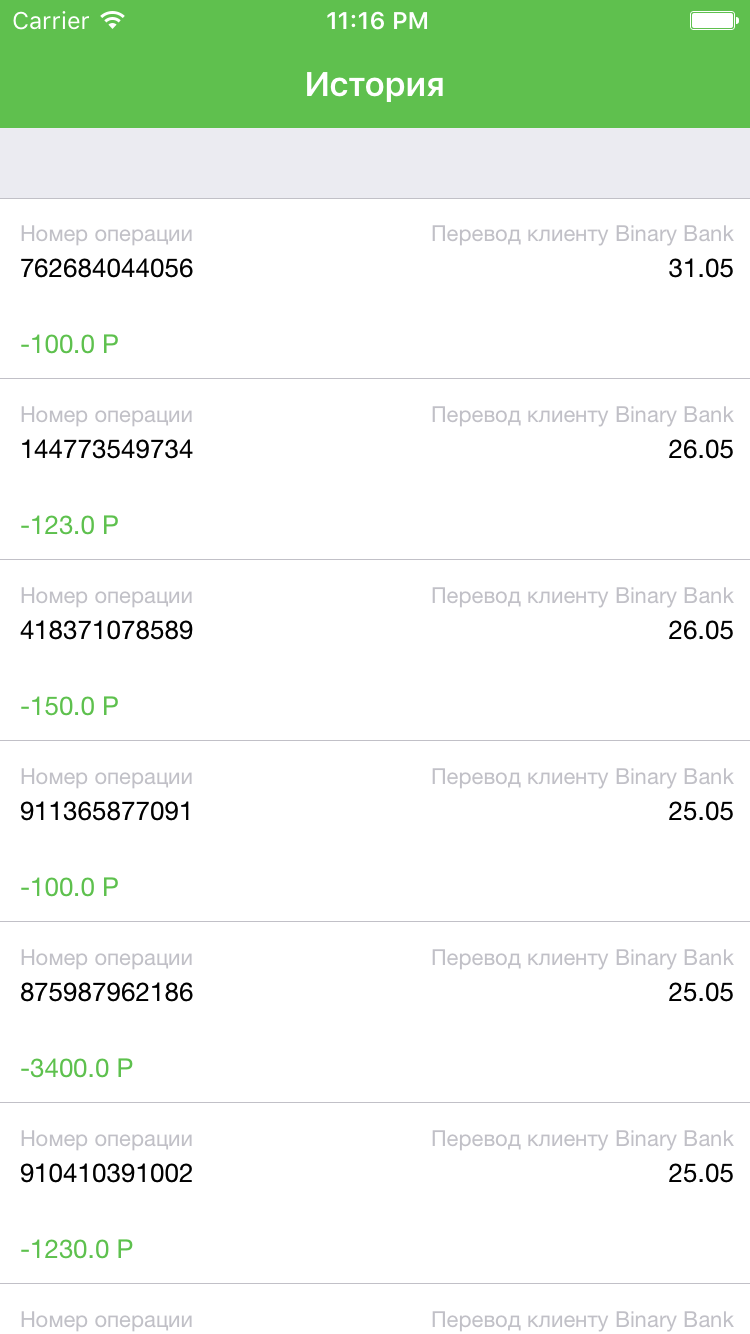
Платежи



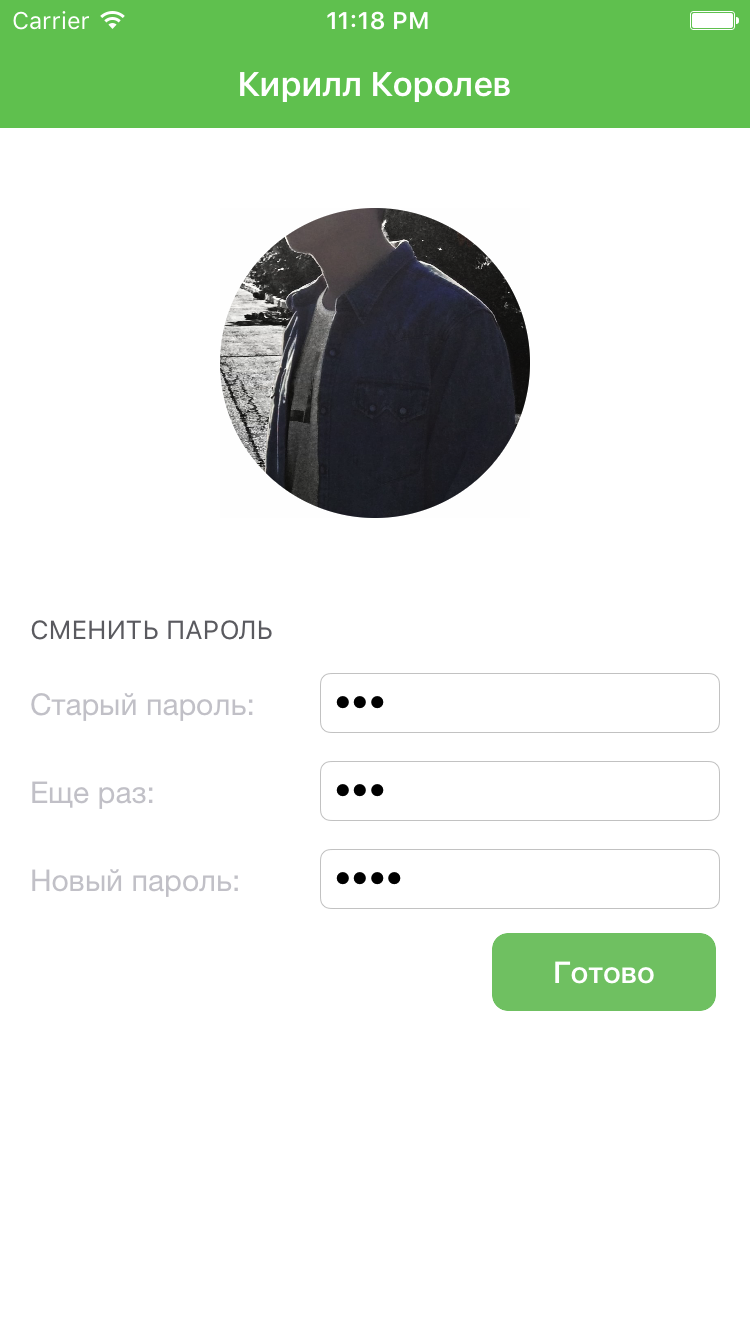


На данной вкладке пользователь может осуществить перевод либо между своими счетами, либо на карту другого клиента, набрав номер его карты.

История

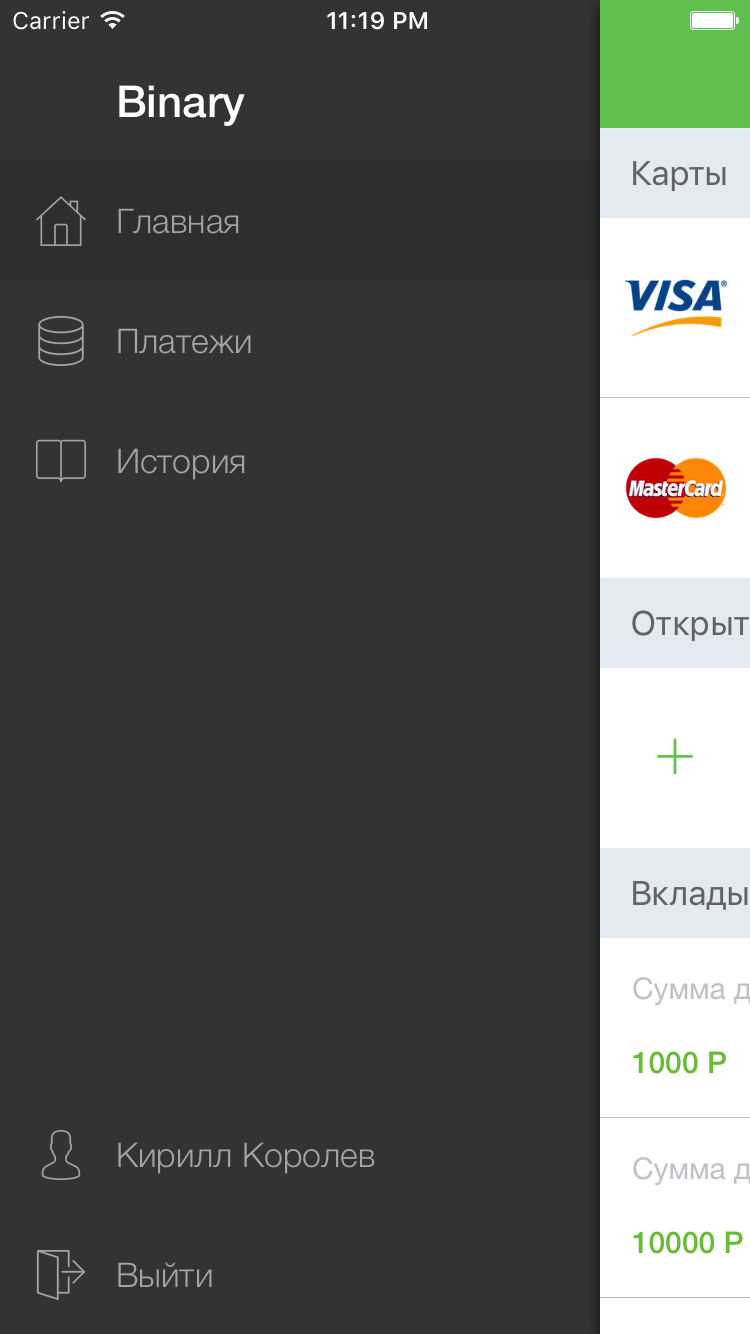
История переводов и платежей по всем картам выводится во вкладке «История»

Личный кабинет



В личном кабинете пользователь имеет возможность сменить пароль.

Меню



Состав классов

Архитектура программы разделена на экраны (Wireframes), в которые входят различные совокупности классов, отвечающие за сущности, отображение и прорисовку, взаимодействие между сервером и так далее.

**Passlock Module**

-PassCode Wireframe

Данный wireframe отвечает за экран ввода пин-кода

**Auth Module**

-SignInButton

-AuthTextField

Данные классы предназначены для создания кастомизированных элементов управления: кнопки входа и текстового поля соответственно.

-User

Сущность, хранящая информацию о пользователе.

-UserManager

Синглтон, выступающий в роли менеджера данных, загружающий информацию о пользователе после ввода логина и пароля.

-AuthWireframe

Wireframe, отвечающий за основую работу экрана авторизации, реализующий протоколы делегатов UserManager и ImageManager.

**Menu Module**

-Menu Wireframe (контроллер, отвечает за взаимодействие объектов на экране «Меню»)

-TopMenuViewController (header-menu, отображающее таблицу)

-BottomMenuViewController (footer-menu, отображающее таблицу)

-Menu Container (контейнер для этих классов)

**Home Module**

-Card (сущность, хранящая информацию о карте)

-Operation (сущность, хранящая информацию об операции)

-Service (сущность, хранящая информацию о сервисе)

-Deposit (сущность, хранящая информацию о депозите)

-Home Wireframe (контроллер, отвечает за взаимодействие объектов на экране «Главная»)

-Home Router (контроллер, отвечающий за навигацию в этом модуле)

-TableViewDelegate (делегат экземпляра класса UITableView)

-HomeTableData (менеджер данных, отвечающий за загрузку данных с сервера)

-HomeTableView (отрисовка view у таблицы)

-Deposit Wireframe (обертка для экрана создания вклада)

-Deposit Container (контроллер для создания вклада)

-CardCell (ячейка таблицы карты)

-DepositCell (ячейка таблицы вклада)

-GoalCell (ячейка таблицы созданного вклада)

**Card Description**

-OperationCell (ячейка таблицы операции)

-CardDescriptionWireframe (контроллер, отвечающий за отображение информации о карте)

-OperationManager (менеджер, отвечающий за загрузку данных об операциях с сервера)

-CardContainerController (модальное окно для показа дополнительной информации о карте)

-DepositManager (менеджер, создающий вклад и отправляющий информацию на сервер)

-DepositLoader (менеджер, загружающий информацию о созданных вкладах)

**Payments Module**

-PaymentsWireframe (контроллер, отвечающий за взаимодействие объектов на вкладке «Платежи»

-PaymentsRouter (навигация по экрану и по его дочерним элементам)

-PaymentsTableDelegate (делегат экземпляра класса UITableView)

-PaymentsCell (ячейка таблицы платежей)

-TransferWireframe (контроллер для перевода клиентам)

-PaymentsView (кастомизированная отрисовка формы платежей)

-SelfTransferWireframe (контроллер для перевода на свои счета и карты)

-SelfTransferCell (ячейка таблицы перевода)

-UploadManager (менеджер для осуществления транзакций)

**History Module**

-HistoryCell (ячейка таблицы истории операций)

-HistoryWireframe (контроллер, который осуществляет взаимодействие между объектам на экране

-HistoryRouter (навигация по экрану и его дочерним элементам)

-HistoryManager (менеджер-синглтон, загружающий информацию об истории)

**Profile Module**

-ProfileWireframe (контроллер, отвечающий за взаимодействие элементов на экране «Личного кабинета»)

-ProfileRouter (навигация по экрану и его дочерним элементам)

-SettingsController (контроллер-контейнер, реализующий логику изменения пароля)

-SettingsManager (менеджер, отправляющий измнененные данные на сервер)

**Router Protocol**

-RouterProtocol (протокол для объектов навигации)

**Supporting Files**

-ProtocolsBundle (набор протокол для различных классов)

-Extensions (расширения классов и полезные фичи)

Реализация backend-a расположена на сервере, к сожалению, доступ к исходникам временно ограничен.

Использованные источники

[*https://stackoverflow.com*](https://stackoverflow.com)

Информация для авторизации

Логин – [kafafyf@gmail.com](mailto:kafafyf@gmail.com)

Пароль - 123