Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

**ОТЧЕТ**

к лабораторной работе

на тему

**ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА**

|  |  |
| --- | --- |
| Студент | Н. С. Слуцкий |
| Руководитель | Е. В. Тушинская |
| Нормоконтролер | Е. В. Тушинская |
| Рецензент | Е. В. Тушинская |

Минск 2023

СОДЕРЖАНИЕ

[Введение 3](#_Toc152511555)

[1 О программном продукте 4](#_Toc152511556)

[2 Тестирование приложения 8](#_Toc152511557)

[2.1 Общие сведения 8](#_Toc152511558)

[2.2 Тестирование базы данных 8](#_Toc152511559)

[2.3 Тестирование некоторых некоторых функций 9](#_Toc152511560)

[2.4 Запуск тестов при загрузке на удалённый репозиторий 10](#_Toc152511561)

[3 Развёртывание приложения 11](#_Toc152511562)

[3.1 Диаграмма развёртывания 11](#_Toc152511563)

[3.2 Развёртывание приложения и другие примечания 11](#_Toc152511564)

[Заключение 13](#_Toc152511565)

[Приложение А (обязательное) Листинг кода 14](#_Toc152511566)

# Введение

Целью данной лабораторной работы ставится:

– разработать полноценное приложение в соответствии с требованиями, изложенными в отчётах лабораторным работам №1 и №2;

– спроектировать приложение таким образом, чтобы пользователю была понятна логика взаимодействия с интерфейсом и в целом задумкой автора;

– развернуть приложение в открытый доступ;

– оформить отчёт о проделанной работе в соответствии со стандартом предприятия БГУИР 2017;

– протестировать работу приложения под разными пользователями и с разных устройств;

– написать тесты для разработанного программного продукта, в которых необходимо протестировать ключевую логику приложения;

– написать краткую инструкцию пользования приложением;

– предоставить диаграмму развёртывания после выкладывания приложения в общий доступ.

# 1 О программном продукте

Разработанный упрощённый аналог виртуальной банковской системы представляет собой полноценное веб-приложение, доступное как с компьютеров, так и с мобильных устройств при наличии браузера и стабильного подключения к интернету.

При заходе на сайт пользователь может наблюдать классическую структуру сайта с блоками навигации и основного содержания. На рисунке 1.1 приведен внешний вид главной страницы сайта разрабатываемой банковской системы.

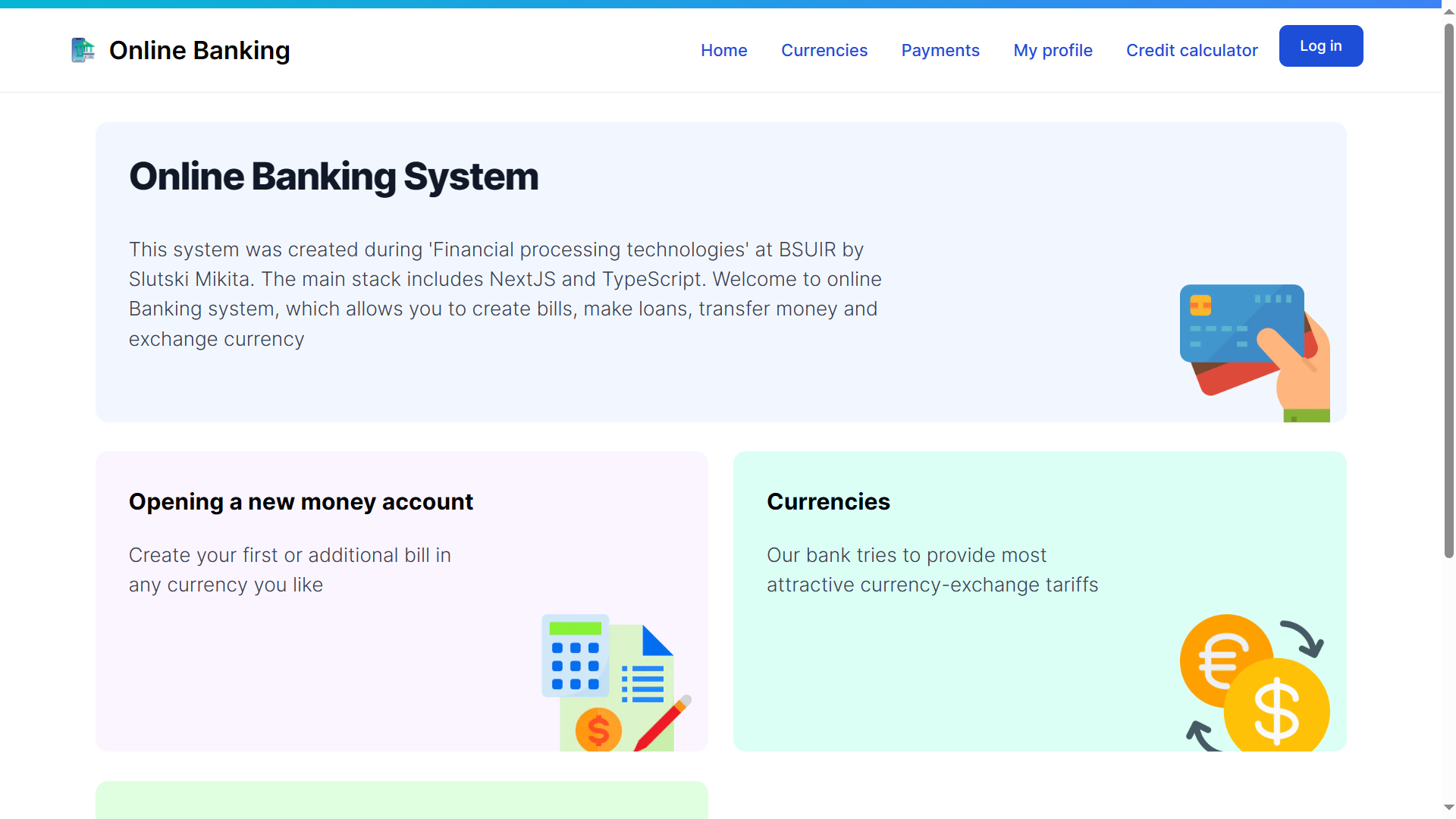


Рисунок 1.1 – Главная страница приложения

Приложение включает в себя следующие разделы:

– главная страница;

– страница входа в приложение и страница регистрации;

– раздел совершения денежных переводов;

– личный кабинет клиента со списком его счетов (виртуальных карт) и созданием заявок на новый счёт;

– страница расчёта месячного платежа для кредита в банке;

– раздел с курсами и конвертером валют.

Все страницы, требующие наличия авторизированного пользователя, соответственно, перенаправляют клиента на страницу входа или регистрации в случае, если клиент на авторизирован. Примерами таких страниц являются раздел с переводом средств или личный кабинет.

В свою очередь, такие разделы, как страница с конвертером валют или страница с кредитным калькулятором, работают и в гостевом режиме и немедленной авторизации не требуют.

На главной странице можно наблюдать промо-блоки с актуальными приложениями банка или просто ссылками на вышеупомянутые разделы.

На рисунке 1.2 приведено изображение страницы с конвертером валют.

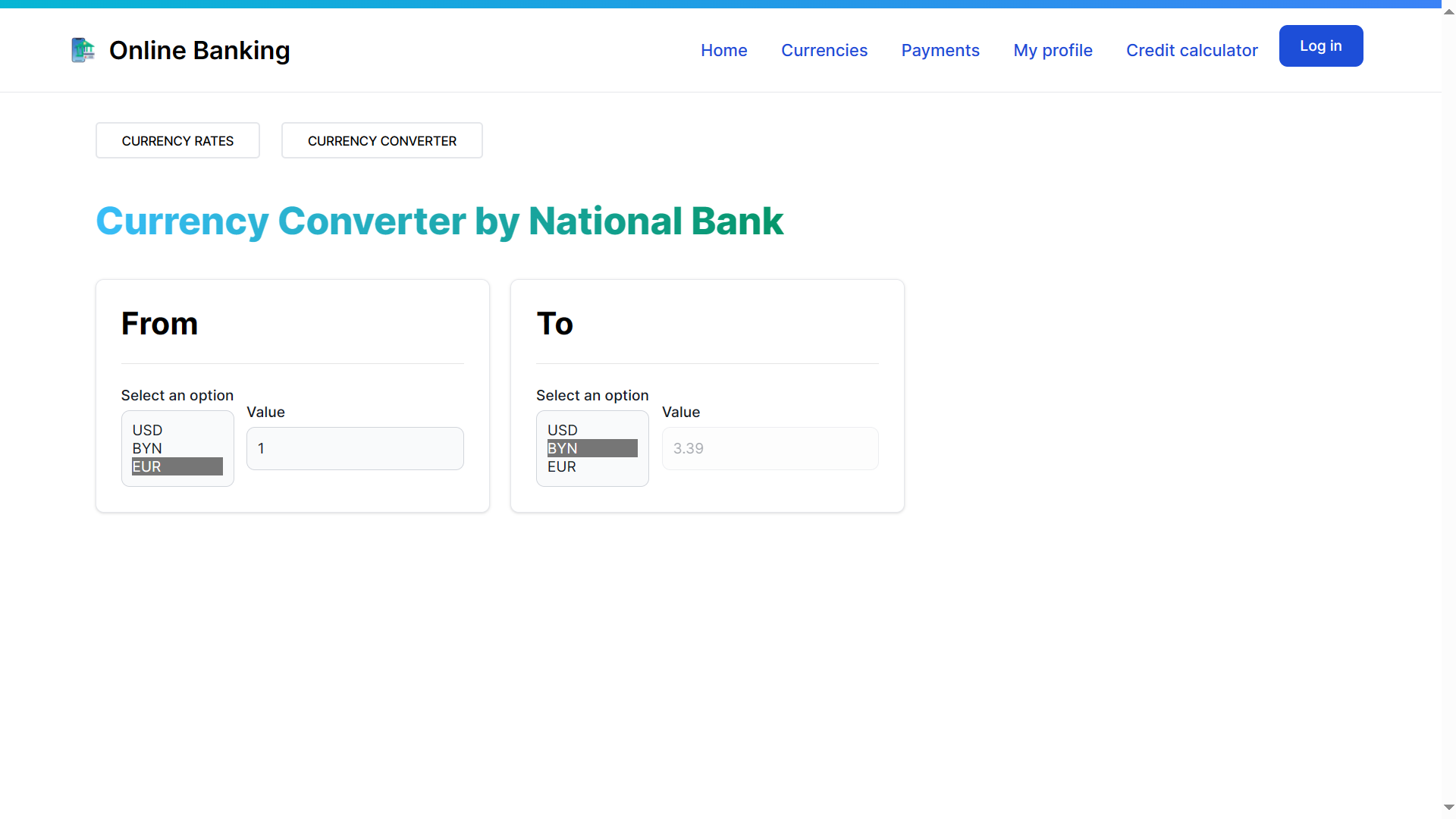


Рисунок 1.2 – Страница с конвертером валют

Страница загружает актуальные курсы валют Национального банка из базы данных. Затем формирует таблицу для быстрых переводов и отображает форму.

Можно выбрать валюту, из которой происходит конвертация, и валюту, в которую происходит конвертация, а также ввести переводимое значение.

Во втором подразделе страницы валют можно ознакомиться с курсами продажи и покупки валюты.

Кредитный калькулятор позволяет быстро рассчитать ежемесячный платёж по кредиту, оценить свои возможности погасить долг без просрочек. На рисунке 1.3 приведён скриншот страницы с реализованным компонентом такого кредитного калькулятора в рамках разрабатываемого приложения банковской системы.

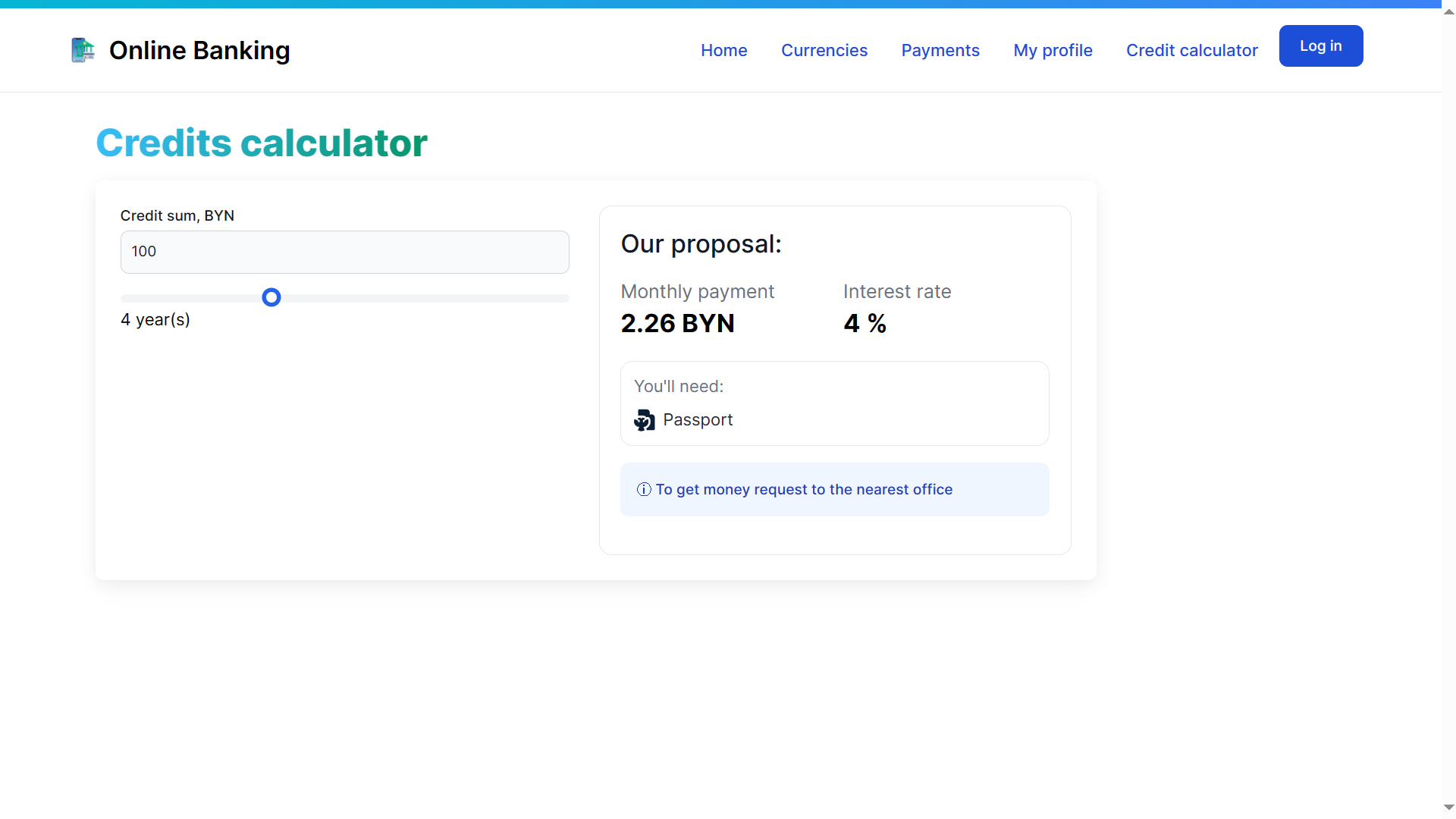


Рисунок 1.3 – Страница с кредитным калькулятором

Клиенту предоставляется возможность открывать разные счета, аналоги виртуальных банковских карт, в любой валюте из предлагаемых на выбор. Через страницу пользователя можно попасть в раздел со списком своих счетов, а, перейдя в карточку счёта, можно увидеть более подробную информацию о нём. Каждый счёт включает валюту счёта, денежный остаток и ассоциированное имя счёта.

Дополнительной опцией предлагается генерация QR-кода, который позволит производить быстрые платежи. В этом QR-коде зашифрован только сам номер счёта. Перейдя на страницу переводов, клиенту предоставляется возможным не вводить вручную номер счёта-получателя, а просканировать его сгенерированный QR-код, чтобы заполнить поле автоматически. Это очень удобная возможность, которая значительно упрощает и ускоряет процесс перевода. Особенно это касается клиентов, которые пользуются приложением с мобильного устройства. Для использования сканера необходимо предоставить браузеру и сайту разрешение использовать встроенную камеру используемого устройства.

При переводе можно выбрать один из своих счетов, с которого будет списана введённая сумма, ввести, собственно, число переводимых денежных единиц и ввести адрес счёта-получателя либо просканировать его QR-код. При проведении платежей между счетами с разными валютами автоматически будет произведена конвертация средств по актуальному курсу банка. На рисунке 1.4 изображён скриншот с мобильного устройства, демонстрирующий возможность использования QR-сканера при проведении быстрых платежей.

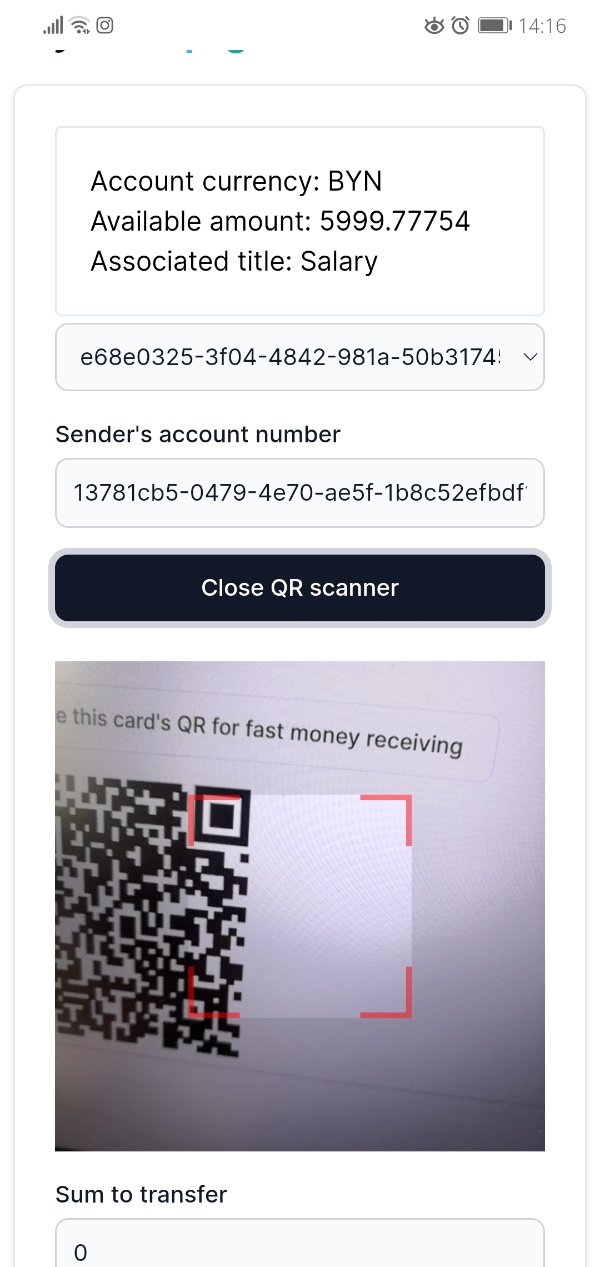


Рисунок 1.4 – Скриншот из мобильной версии сайта из раздела платежей

Про мобильный лэйаут можно сказать, что нигде никакие элементы не вылазят за границы экрана, можно относительно комфортно пользоваться приложением на ходу со смартфона.

# 2 Тестирование приложения

## 2.1 Общие сведения

Для написания тестов приложения используется фреймворк Jest. На данный момент это наиболее распространённое решение для создания тестов приложений в экосистеме NodeJS.

При тестировании целью ставится покрыть тестами основные ключевые части логики приложения. Учитывая, что приложение написано с использованием связки технологий NextJS и Supabase, серверной логики, как таковой, в нём не присутствуют: серверные компоненты напрямую взаимодействуют с базой данных. Поэтому задачей ставится тестирование механизмов Supabase в контексте приложения, тестирование корректности составления с помощью SupabaseJS ORM запросов, созданных триггеров и хранимых процедур.

Для тестирования базы данных необходимо клонировать основную базу данных и производить все тестовые действия на ней. Оригинальная база имеет название OnlineBanking, а база для тестов – OnlineBanking\_\_Test. Данные для таблиц с курсами валют, самими валютами были импортированы из реальной базы данных на момент создания тестов.

Несмотря на описанную концепцию разработки, некоторые важные функции всё-таки реализованы на сервере – поэтому их также необходимо покрыть тестами.

## 2.2 Тестирование базы данных

Тесты запускаются на копии реальной базы данных. Это всё такой же сервер Supabase с запущенным экземпляром PostgreSQL внутри. Перед началом тестов конструкцией beforeAll создаётся клиент Supabase и регистрируются два тестовых пользователя. На каждом шаге проверяется статус операции.

При регистрации пользователь автоматически заносится во встроенную таблицу Auth.Users, однако где-то необходимо хранить и не только стандартные данные, например, имя, фамилию, номер паспорта. Оптимальным решением для создания новой записи в таблице   
Profiles является создание триггера, который после вставки записи во встроенную таблицу Users вытягивает метаданные из запроса и создаёт запись в Profiles.

Первый тест покрывает логику регистрации и работы созданного триггера на таблицу Profiles. Тест проверяет возможность авторизоваться в приложении а также, что триггер отработал – и кастомные данные пользователя также сохранились.

Следующий набор тестов проверяет корректность настройки политик доступа к таблице счетов в базе данных. Проверяется защита доступа неавторизированного пользователя от создания новой записи в этой таблице. Далее последовательно происходит вход в два созданных аккаунта и создание счёта от их имени. Причём также проверяется, что после создания этих счетов у их создателей доступ предоставляется только непосредственно к их собственным счетам.

Одной из важных частей приложения и базы данных является хранимая процедура совершения платежей. В некорректных ситуациях процедура выбрасывает исключение и прерывает транзакцию. Необходимо протестировать функцию как на некорректных, так и на корректных данных. Проверяется факт наличия ошибки при таких случаях, как:

– отрицательная сумма перевода;

– недостаточное количество средств на выбранном счёте отправителя.

После этого совершается запрос с корректными данными и производится проверка на уменьшение средств на счёте-отправителе и увеличение средств на счёте-получателе.

Необходимо проверить механизм входа в приложение. Поэтому несколькими наборами тестируется функция входа. Это могут быть комбинации верного адреса электронной почты, но неверного пароля, или наоборот.

После прохождения тестов необходимо отдельно созданным клиентом Supabase с правами администратора удалить созданные аккаунты клиентов. Это происходит в конструкции afterAll.

## 2.3 Тестирование некоторых некоторых функций

Ввиду наличия системы валидаторов в приложении целесообразным является протестировать эту систему. Тесты покрывают такие функции-валидаторы, как:

– валидатор номера паспорта;

– валидатор адреса электронной почты;

– валидатор номера телефона.

Также тестируется общая функция для валидации целого набора данных как на корректном наборе, так и на невалидном.

Функция, рассчитывающая ежемесячный платёж по кредиту, также покрыта базовым тестом с сайта, с которого была взята формула расчёта кредита.

## 2.4 Запуск тестов при загрузке на удалённый репозиторий

Одним из условий лабораторной работы является выкладывание исходного кода в удалённый репозиторий GitHub. GitHub предоставляет возможность автоматизировать некоторые процессы после каждого сохранения кода в репозитории. Можно разрабатывать сценарии, при которых приложение собирается, тестируется различными средствами и в случае успеха деплоится на сервер.

В случае с разрабатываемым решением с помощью GitHub Actions CI был автоматизирован запуск тестов в репозитории. Таким образом посетитель репозитория может наблюдать, прошла ли текущая выложенная версия приложения тесты или нет. На рисунке 2.1 приведен скриншот результата тестирования внутри GitHub Actions.

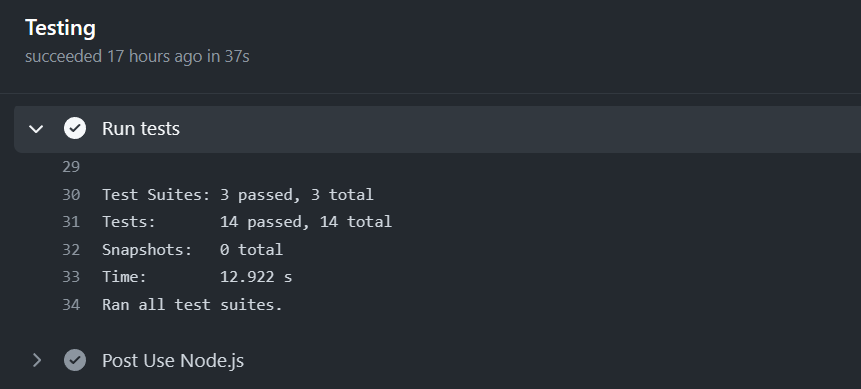


Рисунок 2.1 – Результат удалённого автоматического тестирования

Для написания такого сценария в корне локального репозитория в специальной папке создаётся файлик в формате YAML. В нём описывается используемое окружение и последовательность команд. Для подключения к тестовой базе данных внутри тестов используются секретные ключи. Они должны быть записаны в переменные окружения во время запуска тестов. Локально они хранятся в файлике, который игнорируется системой контроля версий GIT и не попадает на удалённый репозиторий. В репозитории созданы так называемые секретные переменные. Таким образом никакой из ключей доступа не доступен случайному посетителю репозитория.

# 3 Развёртывание приложения

## 3.1 Диаграмма развёртывания

Диаграмма развертывания – это тип UML-диаграммы, которая показывает архитектуру исполнения системы, включая такие узлы, как аппаратные или программные среды исполнения, а также промежуточное программное обеспечение, соединяющее их. Включает в себя следующие элементы: узлы, артефакты, соединения, устройства.

На рисунке 3.1 приведена диаграмма развёртывания для разработанного приложения.

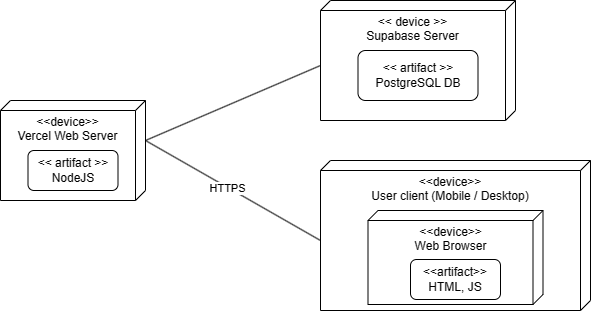


Рисунок 3.1 – Диаграмма развёртывания

Диаграммы развертывания помогают моделировать аппаратную топологию системы по сравнению с другими типами UML-диаграмм, которые в основном описывают логические компоненты системы.

## 3.2 Развёртывание приложения и другие примечания

Приложение разработано на технологии NextJS. Разрабатывающая этот фреймворк компания предлагает удобный способ сборки и автоматического развёртывания – с использованием Vercel CLI.

Vercel – это облачная платформа, специализирующаяся на развертывании статических сайтов и серверных функций. Она позволяет размещать сайты и приложения, которые мгновенно развертываются, автоматически масштабируются и не требуют никаких трудозатрат для обслуживания. Vercel предлагает бесплатный уровень, позволяющий начать работу без внесения каких-либо платежей и допускающий развертывание бессерверных функций для реализации бэкенда

Таким образом приложение может быть выложено в открытый доступ всего за несколько шагов:

– создание аккаунта Vercel;

– установка Vercel CLI;

– авторизация на локальной машине;

– запуск команд vercel и vercel --prod.

Можно настроить GitHub Actions CI на автоматический деплой приложения при успешном прохождении тестов, а можно деплоить вручную. Для этого в файле package.json созданы команды deploy:dev, deploy:prod и   
deploy.

Исходный код приложения можно исследовать в удалённом репозитории GitHub для лабораторных работ по предмету. В репозитории созданы Readme.md файлы, описывающие основные важные моменты, касающиеся проекта, а также инструкции по запуску приложения и тестов.

# Заключение

В результате выполнения был разработан полноценный программный продукт упрощённого аналога банковской системы, а также написаны тесты для ключевой логики приложения. Тесты автоматически запускаются при сохранении кода в удалённом репозитории – и таким образом всегда можно наблюдать актуальное состояние статуса работоспособности ключевых частей программного продукта. В качестве дополнительных видов тестирования для веб-приложения также можно реализовывать:

– тестирование React-компонентов с помощью Testing Library;

– тестирование страниц и пользовательских сценариев с помощью   
Cypress;

– нагрузочное тестирование;

– тестирование безопасности.

Приложение развёрнуто в открытом доступе. А также разработана диаграмма развёртывания и составлен отчёт, оформленный в соответствии с общими требования стандарта предприятия БГУИР.

В приложении А приведена вырезка кода из тестирования базы данных, которая совершает действия перед началом тестирования и после окончания тестирования.

Цели лабораторной работы можно считать достигнутыми.

# Приложение А (обязательное) Листинг кода

beforeAll(async () => {

SupabaseAdmin = createClient(process.env.DB\_URL as string, process.env.DB\_KEY\_ADMIN as string);

Supabase = createClient(process.env.DB\_URL as string, process.env.DB\_KEY as string);

const signUpResponse1 = await Supabase.auth.signUp(user1);

const signUpResponse2 = await Supabase.auth.signUp(user2);

if (signUpResponse1.data.user) {

user1.id = signUpResponse1.data.user.id;

console.log('Signed up a new user: ', user1.id);

}

if (signUpResponse2.data.user) {

user2.id = signUpResponse2.data.user.id;

console.log('Signed up a new user: ', user2.id);

}

});

afterAll(async () => {

const deleteResult1 = await SupabaseAdmin.auth.admin.deleteUser(user1.id!);

expect(deleteResult1.error).toBe(null);

const deleteResult2 = await SupabaseAdmin.auth.admin.deleteUser(user2.id!);

expect(deleteResult2.error).toBe(null);

});