**ГОСТ 19.402-78**

1. **Общие сведения**

**1.1. Наименование и обозначение программы**

Наименование: “ПБК”.

Обозначение: Цифровой сервис подбора банковских карт.

**1.2. Программное обеспечение, необходимое для функционирование программы**

1) Docker;

2) PostgreSQL;

3) ASP.NET Core;

4) Angular.

**1.3. Языки программирования, на которых написана программа**

1. Язык программирования C#;
2. Язык программирования TypeScript.

**2. Функциональное назначение**

**2.1. Назначение платформы**

Сервис подбора банковских карт предназначен для нахождения оптимального банковского продукта (дебетовая карта), используя сведения о расходах пользователя за определенный период времени, а также используемых банковских картах.

Выполняемые функции:

1. Подбор оптимального банковского продукта;
2. Анализ расходов пользователя;
3. Управление данными о расходах пользователя;
4. Разграничение доступа к данным;
5. Управление пользователями.

**2.2. Сведения о функциональных ограничениях на применение.**

Приложение имеет клиент-серверную архитектуру, хранение данных в реляционной базе данных, использование виртуализации в виде контейнеров. Запуск приложения осуществляется путем запуска docker-контейнера.

**3.Описание логической структуры**

**3.1.** **Алгоритм**. Основной алгоритм программы направлен на анализ входных данных и подбор оптимального результата на их основе. Включает в себя следующие сценарии:

-сценарий «Получение списка оптимальных банковских карт через ввод суммы банковских операций»;

-сценарий «Получение списка оптимальных банковских карт через загрузку банковской выписки»;

-сценарий «Заполнение информации о магазине»;

-сценарий «Заполнение информации о банковской карте».

**3.1.1.** **Сценарий «Получение списка оптимальных банковских карт через ввод суммы банковских операций»**.

**Предусловия**: Пользователь авторизован

**Пользовательская история**: Как пользователь я хочу ввести вручную свои банковские операции, чтобы подобрать оптимальную банковскую карту на их основе.

| **Актор** | **Система** |
| --- | --- |
| 1. Пользователь инициирует открытие формы для ввода банковских операций | 2. Система открывает форму |
| 3. Пользователь вводит суммы покупок и магазины, в которых был произведены покупки | 4. Система сохраняет введенные данные в базу данных. |
| 5. Пользователь подтверждает введенные данные. | 6. Система выдает список подходящих банковских карт |

**Альтернативный сценарий №1**: После выполнения шага 4 основного сценария пользователь решает отредактировать введенные данные.

1. Пользователь нажимает “Изменить”.
2. Система открывает форму.
3. Пользователь редактирует введенные данные.
4. Переход на шаг 4 основного сценария.

**Альтернативный сценарий №2**: После выполнения шага 4 основного сценария пользователь решает удалить введенные данные.

1. Пользователь нажимает “Удалить”.
2. Система удаляет введенные данные.
3. Переход на шаг 4 основного сценария.

**3.1.2.** **Сценарий «Получение списка оптимальных банковских карт через загрузку банковской выписки»**.

**Предусловия**: Пользователь авторизован

**Пользовательская истори**я: Как пользователь я хочу загрузить выписку из банка, чтобы подобрать оптимальную банковскую карту на ее основе.

| **Актор** | **Система** |
| --- | --- |
| 1. Пользователь инициирует открытие поля для загрузки банковской выписки | 2. Система открывает поле |
| 3. Пользователь загружает банковскую выписку | 4. Система сохраняет введенный файл в базу данных и отображает пользователю полученные данные. |
| 5. Пользователь подтверждает введенные данные. | 6. Система выдает список подходящих банковских карт |

**Альтернативный сценарий №1**: После выполнения шага 4 основного сценария пользователь решает отредактировать введенные данные.

1. Пользователь нажимает “Изменить”.
2. Система открывает форму.
3. Пользователь редактирует введенные данные.
4. Переход на шаг 5 основного сценария.

**Альтернативный сценарий №2**: После выполнения шага 4 основного сценария пользователь решает удалить введенные данные.

1. Пользователь нажимает “Удалить”.
2. Система удаляет введенные данные.
3. Переход на шаг 5 основного сценария.

**3.1.3. Сценарий «Заполнение информации о магазине».**

**Предусловия**: Пользователь авторизован с ролью Оператор

**Пользовательская история**: Как Оператор я хочу заполнить информацию о магазинах.

| **Актор** | **Система** |
| --- | --- |
| 1.Пользователь инициирует открытие карточки магазина для заполнения | 2.Система открывает карточку |
| 3.Пользователь вводит данные:  -наименование;  -категория;  -список наименований торговых точек;  -MCC. | 4.Система сохраняет введенные данные в базу данных. |
| 5. Пользователь подтверждает введенные данные. | 6.Система сохраняет готовую карточку |

**Альтернативный сценарий №1:** После выполнения шага 4 основного сценария пользователь решает отредактировать введенные данные.

1. Пользователь нажимает “Изменить”.
2. Система открывает карточку.
3. Пользователь редактирует введенные данные.
4. Переход на шаг 4 основного сценария.

**Альтернативный сценарий №2**: После выполнения шага 4 основного сценария пользователь решает удалить введенные данные.

1. Пользователь нажимает “Удалить”.
2. Система удаляет введенные данные.
3. Переход на шаг 5 основного сценария.

**3.1.4. Сценарий «Заполнение информации о банковской карте».**

**Предусловия**: Пользователь авторизован с ролью Оператор

**Пользовательская история**: Как Оператор я хочу заполнить информацию о банковской карте.

| **Актор** | **Система** |
| --- | --- |
| 1.Пользователь инициирует открытие карточки банковской карты для заполнения | 2.Система открывает карточку |
| 3.Пользователь вводит данные:  -наименование банка;  -наименование карты;  -список доступных кэшбек предложений по карте в формате “MCC-доступное предложение”. | 4.Система сохраняет введенные данные в базу данных. |
| 5. Пользователь подтверждает введенные данные. | 6.Система сохраняет готовую карточку |

**Альтернативный сценарий №1:** После выполнения шага 4 основного сценария пользователь решает отредактировать введенные данные.

1. Пользователь нажимает “Изменить”.
2. Система открывает карточку.
3. Пользователь редактирует введенные данные.
4. Переход на шаг 4 основного сценария.

**Альтернативный сценарий №2**: После выполнения шага 4 основного сценария пользователь решает удалить введенные данные.

1. Пользователь нажимает “Удалить”.
2. Система удаляет введенные данные.
3. Переход на шаг 5 основного сценария.

**3.2. Используемые методы.**

-Итеративная разработка. Выполнение работ происходит параллельно с непрерывным анализом результатов и дальнейшей корректировкой последующих этапов работы.

-Model-View-Controller. Программирование с использованием шаблонов проектирования позволяет облегчить работу проектировщиков и разработчиков программного обеспечения. Структурирует код и упрощает тестирование, поддержку: с этим паттерном модернизировать можно отдельные блоки, а не все приложение целиком.

-Экстремальное программирование. Применение данного метода позволяет сделать процесс программирования более гибким.

-Data transfer object. Используется для передачи данных между подсистемами приложения.

-Метод расчета контрольной суммы PBKDF2. Использует псевдослучайную функцию для получения ключей.

-Метод авторизации. Предоставление определённому лицу или группе лиц прав на выполнение определённых действий; а также процесс проверки (подтверждения) данных прав при попытке выполнения этих действий.

-Журналирование. Данная методика позволяет производить запись в хронологическом порядке операций обработки данных.

**3.3. Структура программы.**

Структура программы включает в себя 5 основных частей.

-Frontend;

-Backend;

-База данных;

-Алгоритм для осуществления основного функционала;

-Парсер.

**3.4. Связи программы с другими программами.**

Программа предназначена для функционирования в изолированной системе. Взаимодействие с другими программами и внешними информационными системами не предусмотрено.

**4. Используемые технические средства.**

Для функционирования программы нужны следующие технические средства.

-Поддержка 64-битного ядра и процессора для виртуализации;

-Поддержка виртуализации KVM;

-Не менее 4 ГБ оперативной памяти;

**5. Вызов и загрузка.**

Загрузка компонентов Платформы осуществляется путем выполнения команды ПО Docker **docker compose up** в директории с файлом конфигурации сервисов с именем docker-compose.yml.

После загрузки Платформа переходит в дежурный режим ожидая вызова

запросов от компонент. Вызов осуществляется с помощью согласованных HTTP запросов, содержащие необходимую информацию для начала выполнения заложенных функций программ, путем взаимодействия с API. Запуск клиентской части приложения осуществляется через домен skapex.ru.

**6.Входные данные.**

В качестве входных данных выступают:

-Личные данные пользователя (номер телефона, почта, логин, пароль);

-Наименование торговой точки, сумма в формате “Название торговой точки - сумма осуществленных в нём покупок”;

-Данные магазинов (наименование, категория, список наименований торговых точек, MCC (код, присваиваемый торговой точке платежной системе));

-Данные карт (Наименование банка, наименование карты, список доступных кэшбек предложений по карте в формате “MCC-доступное предложение”).

**7.Выходные данные.**

В ходе функционирования программы будут получены следующие данные:

-Список подходящих пользователю банковских карт в порядке максимального приоритета.

**Глоссарий**

**Docker** - программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений.

**PostgreSQL** - реляционная база данных с открытым кодом.

**ASP.NET Core** - свободно-распространяемый кроссплатформенный фреймворк для создания веб-приложений на платформе .NET с открытым исходным кодом.

**Angular** - фреймворк с открытым исходным кодом. Предназначен для разработки одностраничных приложений.

**C#** - объектно-ориентированный язык программирования общего назначения.

**TypeScript** - расширенная версия языка JavaScript.

**MCC** - код, представляет собой 4-значный номер, классифицирующий вид деятельности торговой точки в операции оплаты по банковским картам в торгово-сервисном предприятии при электронной передаче информации в рамках транзакции за предоставляемые товары или услуги.

**PBKDF2** - стандарт формирования ключа на основе пароля.

**Frontend** - это визуальная часть веб-сайта, которую пользователь видит и с которой может взаимодействовать при помощи браузера.

**Backend** - разработка бизнес-логики продукта. Отвечает за взаимодействие пользователя с внутренними данными

**База данных** - совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных

**Парсер, парсинг** - это алгоритм перевода данных из вида, читаемого человеком в заданную структуру данных.

**Виртуализация KVM** - технология аппаратной виртуализации, позволяющая создать на хост-машине полный виртуальный аналог физического сервера.

**API** - программный интерфейс, то есть описание способов взаимодействия одной компьютерной программы с другими.