### **Название задачи:** Архитектура MVP открытия депозитов

### **Автор:** Чуйкин Павел

### **Дата:** 07.01.25

### **Функциональные требования**

Опишите здесь верхнеуровневые Use Cases. Их нужно оформить в виде таблицы с пошаговым описанием:

| **№** | **Действующие лица или системы** | **Use Case** | **Описание** |
| --- | --- | --- | --- |
| UC1 | Клиенты банка | Подача заявки на депозит online на сайте | 1. Клиент оставляет заявку на сайте 2. Сотрудники бек-офиа вручную обрабатывают заявку 3. Депозит открыт |
| UC2 | Клиенты банка | Подача заявки на депозит online через интернет-банк | 1. Клиент оставляет заявку в приложении 2. Сотрудники бек-офиа вручную обрабатывают заявку 3. Депозит открыт |
| UC3 | Сотрудники бек-офиса | Обработка заявки на депозит | 1. Сотрудник получает заявку на обработку из банка заявок 2. Сотрудник обрабатывает заявку 3. Сотрудник открывает депозит |
| UC4 | Excel таблицы со ставками по депозитам | Миграция в общую базу данных | 1. Ставки по депозитам существуют только в виде таблиц 2. Загружаются в единую систему 3. Ставки по депозитам доступны через интерфейс сотрудникам |
| UC5 | Система онлайн-депозитов | Оповещение клиента о открытии депозита | 1. Сотрудник одобрил заявку на депозит 2. Клиент получил уведомление о успешном открытии депозита |

### **Нефункциональные требования**

Опишите здесь нефункциональные требования и архитектурно-значимые требования.

| **№** | **Требование** |
| --- | --- |
| 1 | Интерфейс интернет-банка и сайта должен быть удобным и соответствовать принятой системе дизайна. |
| 2 | Отклик интерфейсов должен быть минимальным (в пределах миллисекунд). |
| 3 | Сервисы интернет-банка и сайта должны быть доступны 99,9% времени. |
| 4 | Сервисы должны обеспечивать горизонтальное масштабирование. |
| 5 | Система должна выдерживать нагрузку в условиях роста количества заявок. |
| 6 | Запросы в интернет-банке должны обрабатываться равномерно и быстро. |
| 7 | Использовать существующие технологии в банке, включая MS SQL, Oracle и Kafka. |
| 8 | Обеспечить, чтобы системы были совместимы с текущими языками программирования, на которых работает команда. |
| 9 | Предусмотреть разработку документации для дальнейшего масштабирования и поддержки системы. |
| 10 | Планировать переход интернет-банка на микросервисную архитектуру в рамках задачи открытия депозитов. |
| 11 | Интернет-банк должен быть совместим с новыми технологиями (Kafka). |
| 12 | Обеспечить шифрования трафика для передачи данных с сайта и интернет-банка. |
| 13 | Не нужно нагружать базу АБС данными, которые ей не нужны в данный момент |
| 14 | Нужно использовать микросервисную архитектуру при разработке в интернет-банке |
| 15 | Не дорабатывать функционал подрядчика в интернет-банке |
| 16 | АБС поддерживает только вертикальное масштабирование |
| 17 | Следует избегать прямого обмена данными между онлайн-банком и АБС |

### **Решение**

Рис.1 Диаграмма контекстов

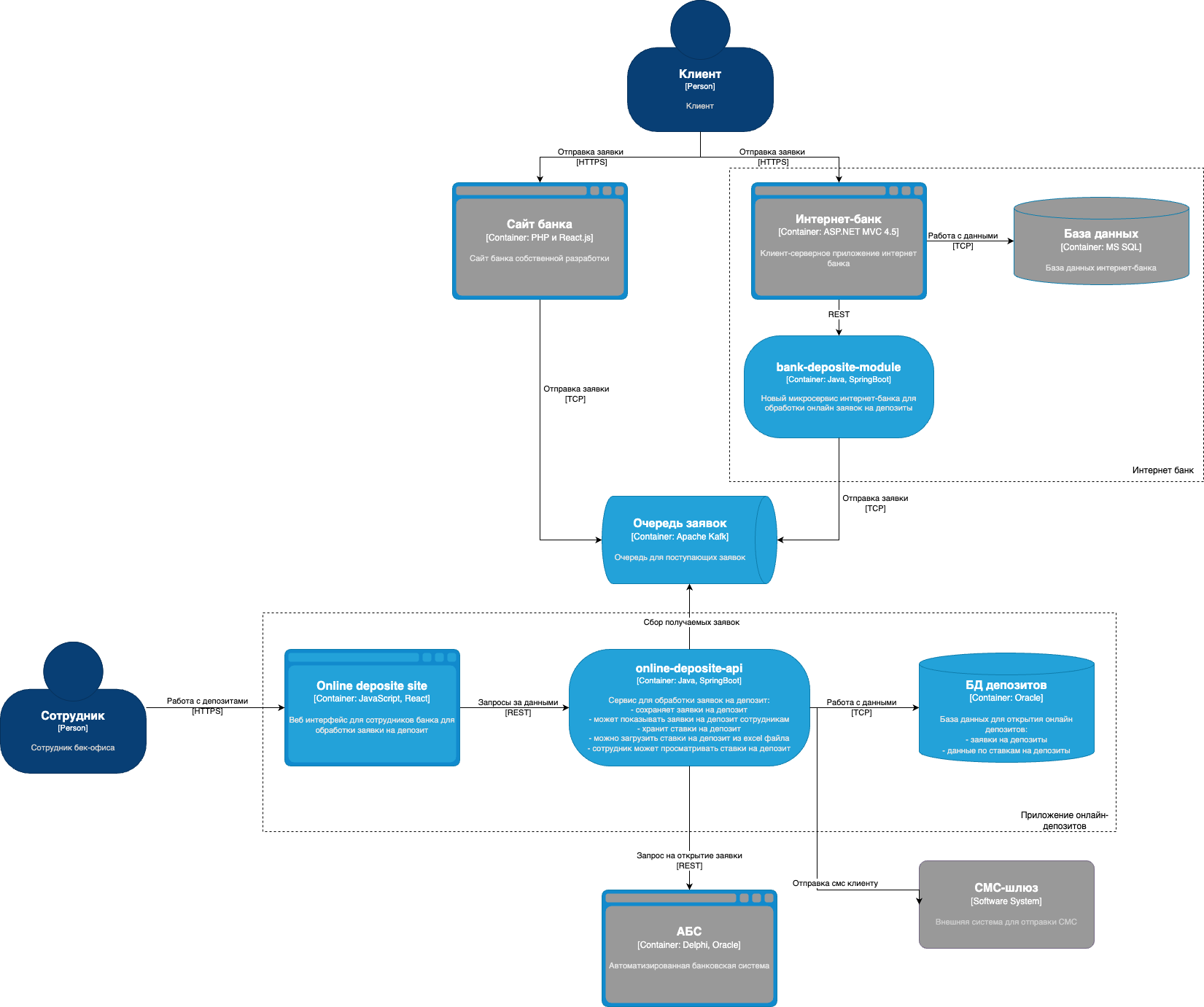


Рис.2 Диаграмма контейнеров

Основным решением является выделение нового приложения, которое отвечает за обработку онлайн заявок на депозиты. Это приложение состоит из 3 компонентов:

* визуальная часть на React, так как в компании есть компетенции по react, а также сайт уже выполнен на этой технологии, так проще будет сохранить стиль и дизайн системы
* бекенд этого приложения - Java, в компании есть компетенции на Java и стоит их развивать на будущее, так как планируется переход на микросервисную архитектуру
* база данных - Oracle, так как в компании уже есть подобная лицензия и экспертиза работы с ней, стоит использовать ее

Функционально приложение будет иметь следующие основные направления:

* получение и сохранения заявки (получение происходит из очереди Kafka, чтобы снизить зависимость ответа клиента от сервиса онлайн заявок и сделать коммуникацию асинхронной)
* отображения заявки сотруднику (происходит через react приложение с взаимодействием с API)
* возможность одобрение заявки сотрудником (происходит через react приложение с взаимодействием с API)
* отправка одобренной заявки в АБС (происходит через react приложение с взаимодействием с API, так как приложение АБС не масштабируемое горизонтально и может не выдержать нагрузки - стоит тут отправлять заявки туда батчами заданного размера в заданные промежутки времени, чтобы не перегрузить стороннюю систему)
* отправка СМС при успешной отправки заявки (после того, как АБС ответил положительно о открытии депозита, идет запрос на отправку СМС, для персистентности тут можно использовать паттерн OutBox)
* загрузка данных о ставка по депозиту (сейчас данные находятся в excel таблицах, нужна возможность их загрузить в систему, чтобы сотрудники могли их видеть из единого источника)

Также для функционирования внедряется очередь сообщений Kafka, которая будет в себе аккумулировать заявки, в том числе это нужно, как опыт использования для дальнейшего перехода на микросервисную архитектуру.

Доработки внешних систем:

* нужно добавить возможность клиенту отправлять из сайта заявку (PHP/React) с нужными полями в Kafka
* в интернет банке нужно также добавить возможность отправлять данные в Kafka, для этого стоит выделить отдельный сервис, который будет этим заниматься
* в системе АБС нужно будет предусмотреть возможность изменения статуса сразу нескольких заявок

Дополнительно:

* во всех системах стоит использовать TLS/SSL шифрование для передачи данных клиента, чтобы обеспечить безопасность
* при разработки тщательно документировать как контракты внешнего API, так и внутреннюю архитектуру
* по всем важным решениям стоит вести ADR в каждом проекте, чтобы можно было восстановить логику принятых решений
* разработку новых форм и функционала в интернет банке и на сайте стоит вести в соответствии с текущей дизайн системой
* систему сайта нужно сделать резервируемой и автоматически переключаемой на запасную ноду
* при создании топиков/партиций в Kafka предусмотреть, что АПИ онлайн заявок будет развернуто в нескольких экземплярах

### **Альтернативы**

* отказ от использования Kafka, использования синхронных запросов - в таком случае скорость ответа для клиентов будет сильно зависеть от поведения сервиса онлайн заявок на депозиты
* отказ в целом от отдельного сервиса и реализация данного функционала внутри монолита АБС - слишком проблематично с точки зрения нагрузки, так как АБС плохо масштабируется, а так же чревато переусложнением кодовой базы и замедлением дальнейшей разработки

**Недостатки, ограничения, риски**

* на данном этапе АБС все еще сдерживает возможности производительности системы, стоит обратить на это внимание и возможно усовершенствовать эту систему
* внедрение новой технологии - Kafka - опасно возможными проблемами из-за недостатка компетенций