Слайд 2:

Определенным бизнесам требуются специфичные CRM системы для разных сфер деятельности. Например, универсальным банкам требуется отслеживать отношения с клиентами в разных сферах деятельности: различные виды кредитования (авто, потребительские, ипотека), инвестиционные, депозитарные и множество других. Проблема заключается в том, что готовые CRM-системы не могут полностью соответствовать всем внутренним требованиям организации. Это и технические, как полная интеграцию в инфраструктуру организации или соответствие всем мерам информационной безопасности, и требованиям к бизнес-процессам, например интеграция с СППР. Поэтому банкам удобно разрабатывать своё ПО для данных задач.

Также очень важный фактор это - уход крупных IT организаций с рынка РФ. Например IBM и SAP, которые поставляли свои CRM и другие системы.

Слайд 3:

Прохожусь по списку, тут нечего добавить

Слайд 3а:

Говорю, что создана ментальная карта с планом ВКР

Слайд 4:

CRM-системы делятся на четыре основных типа:

* Стратегические CRM – это ориентированные на клиента системы, нацеленные на привлечение и удержание прибыльных клиентов. Эти системы собирают и разделяют данные о клиентах, которые в свою очередь используются совместно с информацией о рыночных трендах, чтобы создавать более привлекательные предложения для клиентов.
* Операционные CRM основаны на таких функциях как продажа, маркетинг и обслуживание клиентов. Такие CRM занимаются автоматизацией этих процессов. Для них свойственен дэшбоард, который дает общее представление о всех трех функциях на одной панели для каждого клиента.
* Аналитические CRM основаны на сборе, интерпретации, разделении, хранении, изменении, обработке и представлении данных, связанных с клиентом.
* Коллаборативные CRM занимаются созданием платформы для объединения внешних заинтересованных сторон, таких как: поставщики, продавцы и дистрибьюторы, и обменом информацией о клиентах между ними. Такая система используется в сценарии B2B, где множество бизнесов может совместно заниматься разработкой продуктов, исследованиями рынка и маркетингом.

Слайд 5:

CRM-системы в своём корне состоят из трех основных компонентов, это:

1) Хранилище данных представляет их себя базу данных, где хранятся все данные, используемые в CRM-системе. База данных может быть как реляционной, так и не-реляционной. Выбор полностью зависит от технических решений и задач CRM.

2) Прослойка бизнес-логики обычно представляет из себя промежуточное звено между хранилищем данных и интерфейсом. Это сервис, который получает данные из хранилища данных, обрабатывает их и отправляет в интерфейс, где пользователь уже взаимодействует с ними.

3) Графический интерфейс обычно представляется в двух видах: web-интерфейс и desktop-интерфейс. Web-интерфейс отображается доступен из браузера с любого устройства, когда desktop-интерфейс требует установки и настройки на ПК. Интерфейс является компонентом взаимодействия пользователя и системы. Также интерфейс предоставляет дополнительную безопасность системе, например: ограничивает пользователю использование определенных функций или проверяет вводимые данные.

Слайд 6:

На данном слайде изображена схема, которая отображает архитектуру нашего решения.

За основу мы взяли мкросервисную архитектуру. Это решение было принято по следующим причинам:

* В микросервисной архитектуре компоненты системы могут дорабатываться и выноситься на продукционное окружение независимо друг от друга;
* Компоненты в данной архитектуре являются простыми и портативными, что позволяет их легко и быстро дорабатывать и масштабировать.

Стандартные микросервисы в нашей системе будут разработаны по шаблону проектирования MVC с применением шаблона Repository Pattern.

Интерфейс или же frontend, представляет из себя SPA приложение с применением гибридным применением SSR и CSR.

База данных будет реляционной так как такие БД идеальны для хранения типизированных строго-структурированных данных.

Общение backend и frontend будет происходить посредством RESTful API.

В свою очередь backend будет взаимодействовать с БД посредством механизмов, реализованных ORM.

О всех элементах мы поговорим поподробнее в следующих слайдах.

Слайд 7:

Наша система включает в себя следующие механизмы ИБ:

* Идентификация — сохранение записи пользователя в системе с его данными и ролями. Позволяет использовать следующие два механизма;
* Аутентификация — процедура проверки подлинности, например, проверка подлинности пользователя путем сравнения введенного им пароля с паролем, сохраненным в базе данных;
* Авторизация — предоставление определенному лицу или группе лиц прав на выполнение определенных действий;
* Cross-Origin Resource Sharing (CORS) — кратко, это механизм позволяющий указать из каких источников и какие запросы могут приходить к backend. Все запросы, которые происходят из неуказанных источников или с запрещенными методами или заголовками отклоняются.
* Валидация – проверка введенных пользователями данных, если данные не прошли валидацию, то они не сохраняются.

Слайд 8:

Тут я перечисляю технологии и все такое. Ну и говорю, что их на самом деле дофига, а это прям самые фундаментальные.

Слайд 9:

Выбор БД представляет из себя более объективное решение. Так как они имеют конкретные критерии, то можно сравнить их. Мы использовали МАИ с девятибалльной шкалой, где чем больше число, тем предпочтительней альтернатива.

Сравнение происходило по следующим критериям: …. В итоге PostgreSQL оказался наиболее предпочтительным для нас вариантом.

Слайд 10:

Поговорим немного про backend. Как уже было сказано, backend это микросервисы, разработанные по шаблону MVC. Для реализации данного шаблона мы испольщзовали модуль Spring – Spring MVC.

Так же мы применили шаблон Repository Pattern. Данный шаблон проектирования позволяет полностью абстрагироваться от низлежащей базы данных и способов взаимодействия с ней, позволяя использовать модели и репозитории. Данный шаблон реализован с помощью. Spring Data JPA.

Spring Data JPA использует под капотом Hibernate ORM и полностью соответствует принципу Persistence Ignorance, который гласит, что классы, моделирующие бизнес-домен в системе, не должны зависеть от того, как они могут храниться. Таким образом, их дизайн должен максимально точно отражать идеальный дизайн, необходимый для решения текущей бизнес-задачи, и не должен быть изменен проблемами, связанными с тем, как состояние объектов сохраняется и впоследствии извлекается.

Слайд 10а:

Тут своими словами, я и так это раз 100 рассказывал.

Слайд 10б:

То же самое

Слайд 11:

И тут родной язык

Слайд 12:

Перед разработкой мы составили ТЗ для трех модулей системы:

* Clients – модуль для работы с данными о клиенте
* Loans – модуль для работы с данными о кредитах клиента
* Cars – модуль для работы с данными о машинах клиента

ТЗ для каждого модуля включает в себя необходимые поля, типы данных, типы полей, обязательность этих полей и правила валидации.

Слайд 12а:

Так же мы составили UML схему, в которой представили структуру программного кода. Тут я прохожусь по компонентам.

Слайд 13:

После написания ТЗ мы определили организацию разработки. Мы использовали Agile с итерацией каждую неделю и отслеживали задачи на канбан доске.

Слайд 14 15 16:

Просто показываю че за компоненты и все такое

Слайд 17:

Перечислю все, что мы сделали и говорю про полезность нашего решения.