# Базовая концепция архитектуры

Требовалось создать платформу с динамической структурой данных, которая, к тому же, будет изменяться в процессе эксплуатации, притом на "горячую", т.е. нужна возможность изменять структуру данных во время работы с этими данными, т.е. добавления/изменения/удаления оных. И, при этом, обеспечивать высокую нагрузку.

Логически платформа представляет набор бизнес-объектов (БО), у которых есть поля различных типов. У каждого бизнес-объекта можно создавать инстанции - данные. Значения полей инстанций в последствии можно менять. Инстанции могут быть помечены как:

1) Актуальные - первичное состояние после создания. В этом состоянии инстанция находиться при работе с ней.

2) Удалённые - такие инстанции считаются удалёнными - они не участвуют в работе системы. Так же есть возможность удалить физически инстанции.

3) Архивные - давно не используемые инстанции, ушедшие в архив.

4) Тестовые - инстанции, которые используют для тестирования системы.

Для этого были выбраны системы управления баз данных, которые поддерживают масштабирование "в ширину". А именно:

1) MongoDB

2) Apache Kafka

3) ElasticSearch

4) PostgreSQL

5) Apache Zookeeper

## MongoDB - главное оперативное хранилище

Главной базой данных является MongoDB. В ней хранятся все данные системы в оперативном режиме.

В оперативном режиме - это значит, что, например, если у клиента фамилия поменялась с "Иванова" на "Сидорова", то в MongoDB остаётся только "Сидорова", а "Иванова" стирается бесследно.

Более того, база данных MongoDB используется только в режиме Key-Value-Storage, это обозначает, что платформа в неё делает запросы только по идентификаторам с ожиданием единичного документа по этому идентификатору.

На платформе часто требуется получить списки - использовать для этого MongoDB запрещено, чтобы не нагружать её. Для этого используется ElasticSearch.

## ElasticSearch - получение информации списками

ElasticSearch используется для получения оперативной информации в виде списков, которые отфильтрованы и отсортированы по желанию пользователя в режиме постраничной загрузки.

В ElasticSearch тоже хранится только оперативная информация.

Для того чтобы синхронизировать данные в ElasticSearch и MongoDB используется ApacheKafka.

## Apache Kafka - синхронизация данных + хранение истории изменений данных

Apache Kafka на платформе MyBPM используется для двух целей:

1) Хранение исторической информации;

2) Синхронизация данных между различными компонентами системы.

Все изменения в данных системы сбрасывается в Kafka в виде дельты данных. В дельте данных содержится только та информация, которая изменилась в данный момент в системе. А именно, например, если у клиента сорок полей, а изменилось только фамилия, то в дельту попадёт только эта фамилия, при том, только новое значение этого поля. Так же в дельте будут координаты того, что изменилось - идентификатор инстанции. А также кто и когда это изменил. Резюмируя сказанное, в дельте содержится следующая информацию:

1) Новые значения полей, которые изменились

2) Координаты того, что изменилось: идентификатор инстанции бизнес-объекта + идентификатор самого бизнес-объекта

3) Идентификатор пользователя, кто сделал это изменение

4) Дата и время этого изменения

Дальше Kafka-Consumer-ы подгружают эти изменения и применяют их для соответствующих компонентов системы. Например, соответствующий Consumer обновляет данные в ElasticSearch тем самым синхронизируя их с оригиналом в MongoDB.

## Apache Kafka - как база исторических данных

К Apache Kafka следует относиться как к базе данных, в которой храниться история всех изменений системы.

На платформе Apache Kafka организована динамическая система получения отчётов.

Вначале пользователь подготавливает структуру отчёта и сохраняет её в БД MongoDB. Далее платформа создаёт в PostgreSQL таблицы для получения данного отчёта, но эти таблицы пустые. Так как в Apache Kafka храниться вся история изменений, то можно пройтись по ним и подготовить данные в этих таблицах, что и делается. Эти таблицы заполняются и отчёт готов для эксплуатации. Далее Kafka-Consumer отслеживает дальнейшие изменения и держит таблицы в актуальном состоянии.

Для того чтобы данная функциональность работала, необходимо исторически данные держать постоянно в Apache Kafka и не стирать их. В Apache Kafka для это есть весь необходимый инструментарий - репликация, партиции и прочее доступные из коробки.

Следует отметить, что платформа MyBPM позволяет разделять данные инстанций БО на разные Kafka Topic-и. Это позволяет зачищать те БО, которые сильно растут в размере. При этом по ним теряется историческая информация. А это означает, что по ним не должно быть отчётов с исторической информацией.

## Предназначение PostgreSQL

PostgreSQL используется для решения сложных логических задач. Например, в PostgreSQL хранятся ловушки событий, которые срабатывают, если наступает заранее ожидаемое событие, например ожидание сохранение определённой инстанции БО, чтобы запустился конкретный скрипт в бизнес-процессе.

## Предназначение Apache Zookeeper

В Apache Zookeeper платформа MyBPM хранить всю конфигурационную информацию - параметры доступа, таймауты, размеры буферов и прочее.

В дальнейшем планируется перенести всю эту конфигурацию в БД MongoDB и управлять ей через интерфейс.

# Реализация платформы MyBPM

Платформа MyBPM состоит из двух программ, оформленных в виде докер-образов.

1) Клиентская часть - WEB : out.mybpm.kz/mybpm-web-...

2) Серверная часть - API : out.mybpm.kz/mybpm-api-...

## Клиентская часть - WEB

WEB-программа написана с использованием свободной библиотеки Angular от Google с открытыми исходными кодами.

Оформлена она в докер-образе под управлением сервера приложений NGINX.

## Серверная часть - API

API-программа написана на языке программирования Java с использованием библиотеки Spring-Boot.

Оформлена она в докер-образе под управлением сервера приложений из Spring-Boot на базе Apache Tomcat.

# Схема работы платформы MyBPM



(Далее номера в круглых скобках обозначают индексы на картинке выше)

## Основной алгоритм внесения изменений на платформе

Если нужно изменить данные на платформе, то срабатывает следующая последовательность:

1. Платформа делает изменение в MongoDB.
2. Если это удалось,
3. То записывается данное изменение в Kafka.
4. Дальше срабатывают все Kafka-Consumer-ы, которые отслеживают это изменение, и синхронизируют другие компоненты системы.

Желательно Kafka-Consumer-ы выносить на отдельные сервера, как показано на рисунке под пунктом (7).

## Описание работы платформы MyBPM

Пользователь (1) открывает браузер (2) и указывает адрес Web-компонента (3) платформы MyBPM. Браузер загружает HTML и JS с Web-компонента и запускает работу клиентской программы на языке JavaScript, который собран с помощью библиотеки Angular от корпорации Google.

Web-компонент (3) не содержит в себе никакого состояния (Stateless) и поэтому можно запустить несколько его инстанций на случай если одна не будет справляться с нагрузкой.

Пользователь (1) видит интерфейс в браузере (2) и начинает работать с системой. Чтобы система смогла получить данные, клиентская часть (3) вызывает Rest-запросы к компоненту API (4) по Web API, который предоставляет компонент API как показано на рисунке выше.

Компонент API (4) - это приложение на языке Java с использованием библиотеки Spring Boot, которое предоставляет различные сервисы для окружающих компонентов (от сюда название API).

Компонент API (4) не содержит в себе никакого состояния (Stateless) поэтому можно запустить несколько его инстанций, на случай если одна не справляется с нагрузкой.

Так как компонент API (4) не содержит в себе состояние, то ему необходимы базы данных, где это состояние будет храниться. К компоненту API (4) необходимо подключить пять типов баз данных: MongoDB, Kafka, ElasticSearch, Zookeeper, PostgreSQL.

С платформой работают внешние сторонние системы (6) для этого на платформе предусмотрен внешний API (5).

# Структурная схема MyBPM с отделённым хранилищем файлов





## Описание структурной схемы

Данная схема применяется в том случае, когда на платформе предполагается использовать большое количество файлов.

# Требования к настройке баз данных со стороны платформы MyBPM

[Смотрите по ссылке](../001_install/030-MyBPM-API-DB-require.md)