Архитектура и инструменты избранных случаев интеграций по данным (часть 2)

Владимир Красильщик
JUG Ru Group

Что было в первой части доклада?

15% разработка компонента

85% межкомпонентное взаимодействие

Что было в первой части доклада?

Что было в первой части доклада?

1. Понятия master system, golden source, интеграция по данным

Что было в первой части доклада?

1. Понятия master system, golden source, интеграция по данным

2. Характер и объем данных влияет на API, см. матрицу интеграции по данным

Что было в первой части доклада?

1. Понятия master system, golden source, интеграция по данным

2. Характер и объем данных влияет на API, см. матрицу интеграции по данным

3. Кейз 1: подход с выгрузками в общий S3, ClickHouse, YT

Что было в первой части доклада?

1. Понятия master system, golden source, интеграция по данным

2. Характер и объем данных влияет на API, см. матрицу интеграции по данным

3. Кейз 1: подход с выгрузками в общий S3, ClickHouse, YT

4. Кейз 2: debezium для рефакторинга dblink

Что было в первой части доклада?

1. Понятия master system, golden source, интеграция по данным

2. Характер и объем данных влияет на API, см. матрицу интеграции по данным

3. Кейз 1: подход с выгрузками в общий S3, ClickHouse, YT

4. Кейз 2: debezium для рефакторинга dblink

5. Кейз 3: StreamSets для наполнения озера данных

Что было в первой части доклада?

1. Понятия master system, golden source, интеграция по данным

2. Характер и объем данных влияет на API, см. матрицу интеграции по данным

3. Кейз 1: подход с выгрузками в общий S3, ClickHouse, YT

4. Кейз 2: debezium для рефакторинга dblink

5. Кейз 3: StreamSets для наполнения озера данных

6. Подход “Shared Database” и доступ по jdbc в витрину на реплике

Что было в первой части доклада?

1. Понятия master system, golden source, интеграция по данным

2. Характер и объем данных влияет на API, см. матрицу интеграции по данным

3. Кейз 1: подход с выгрузками в общий S3, ClickHouse, YT

4. Кейз 2: debezium для рефакторинга dblink

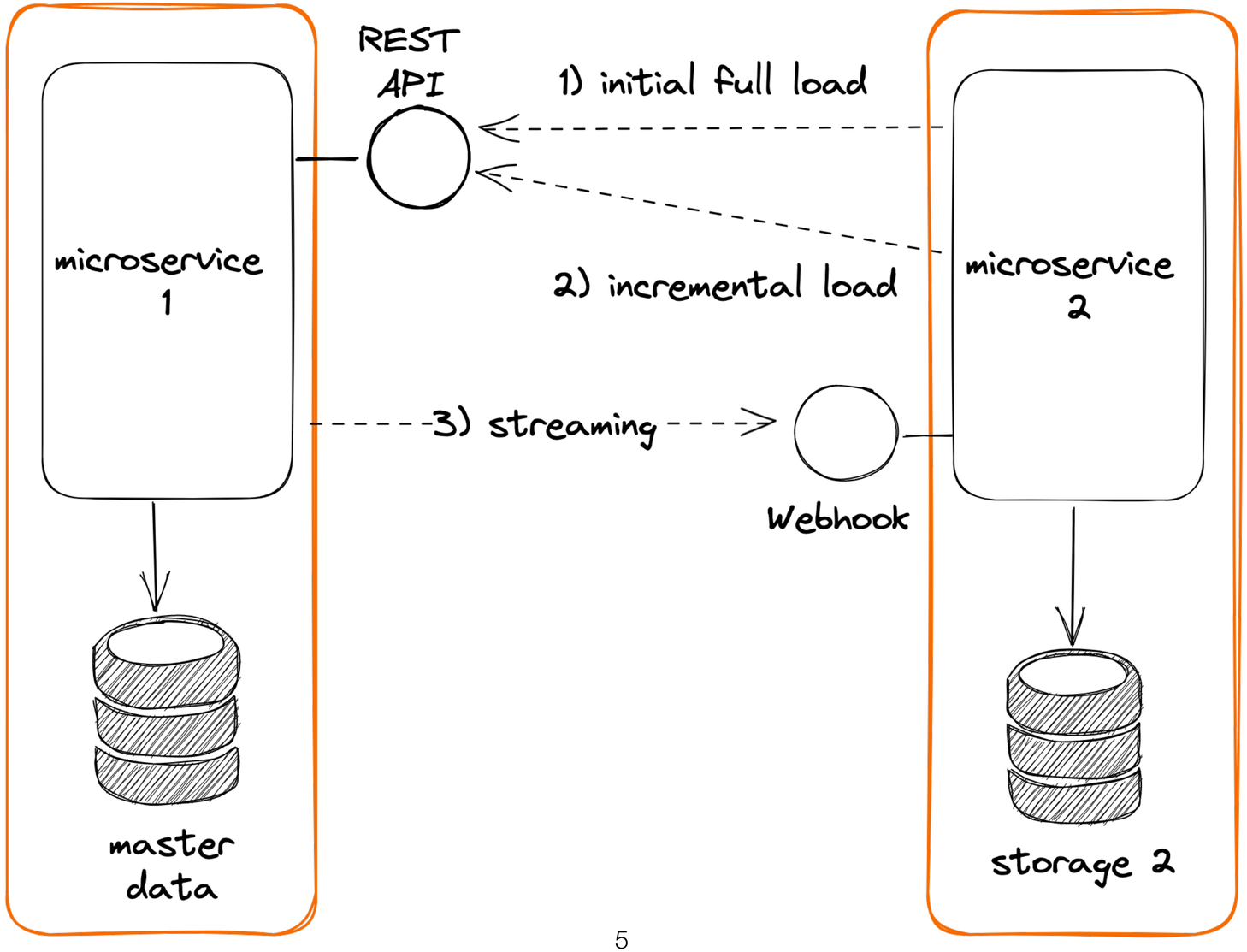
5. Кейз 3: StreamSets для наполнения озера данных

6. Подход “Shared Database” и доступ по jdbc в витрину на реплике

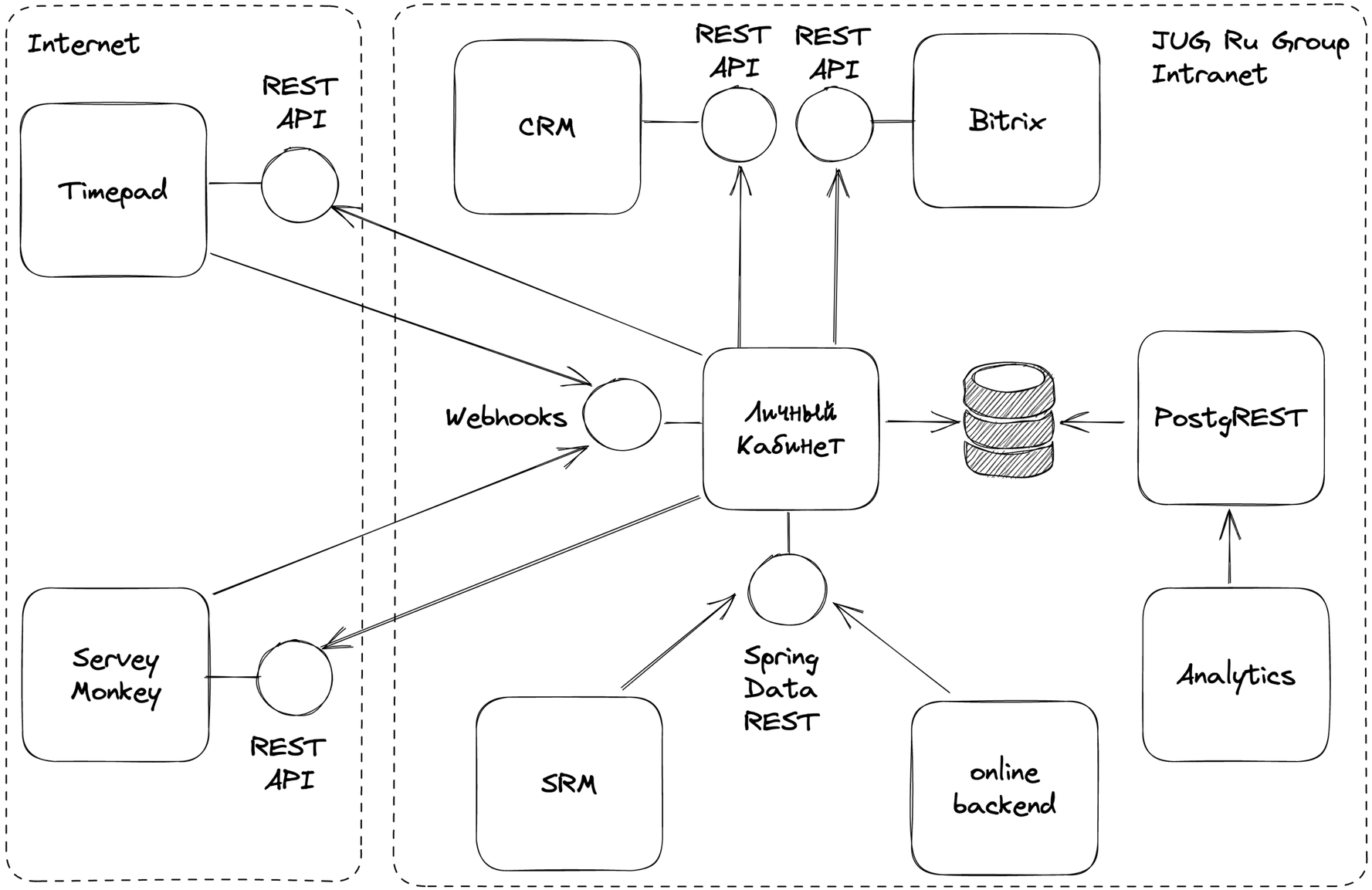
7. Аудит данных, реализация на тригерах в БД

Как еще достать или выставить данные? REST API!

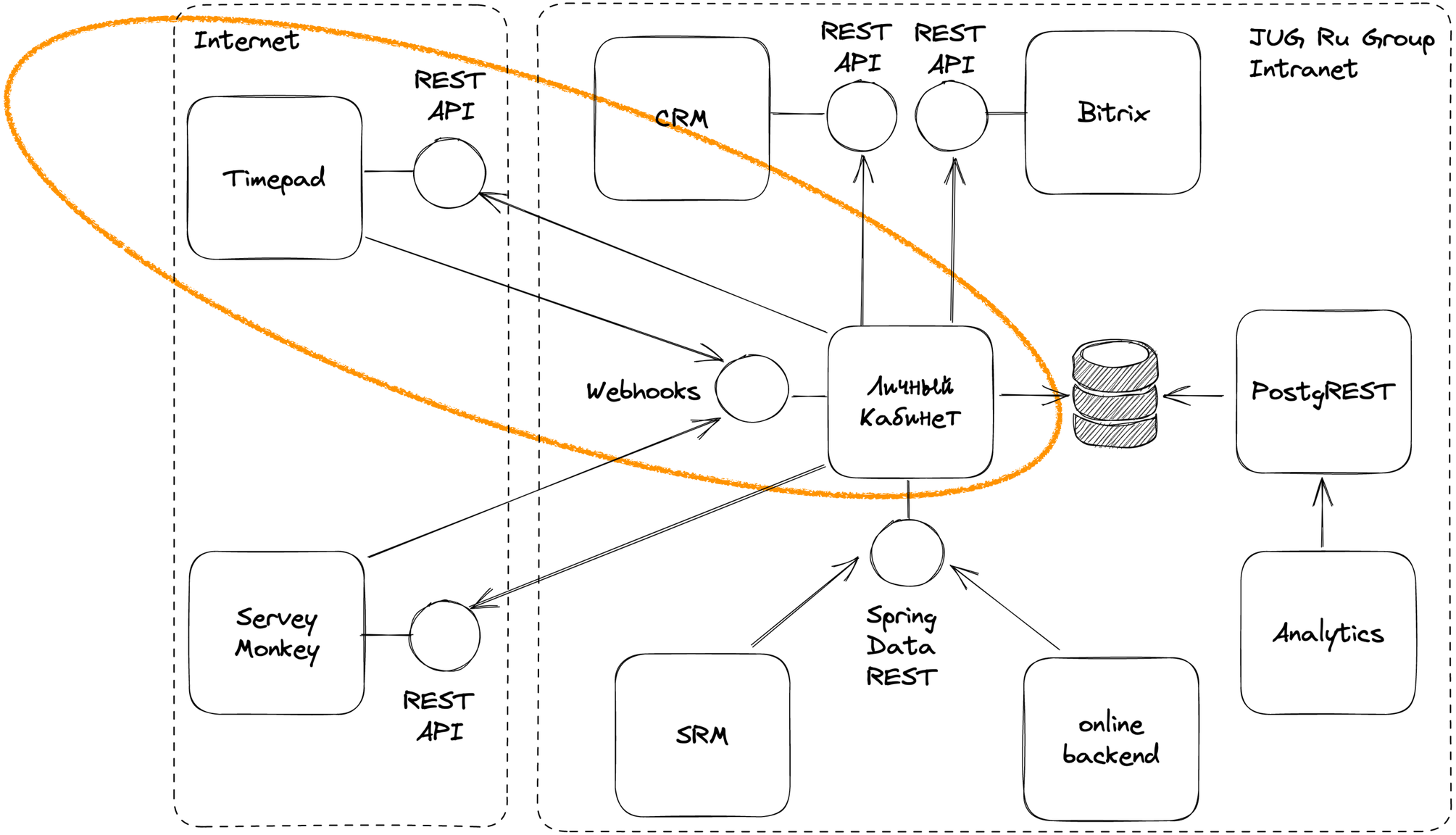
Как еще достать или выставить данные? REST API!



Кейз №4


Internet
REST API
Timepad
Servey Monkey
REST API
CRM
REST API
REST API
Bitrix
JUG Ru Group Intranet
Webhooks
Личный Кабинет
PostgREST
Spring Data REST
SRM
online backend
Analytics

Кейз № 4.1


Internet
REST API
Timepad
Servey Monkey
REST API
CRM
REST API
REST API
Bitrix
JUG Ru Group Intranet
Webhooks
Личный Кабинет
PostgREST
Spring Data REST
SRM
online backend
Analytics

Как еще достать или выставить данные? REST API!

```
@RequestMapping(value = "/restws/v1/orders", method = GET)
public RestWsResponseView getOrders(
@RequestBody GetOrdersRequestView view) {
    return orderService
            .getOrders(
                    view.getFilter().getModifiedAtStartingFrom(), // -- 1
                    view.getPaging().getPageNumber(), // -- 2
                    view.getPaging().getPageSize()) // -- 3
            .mapElements(OrderView::new)
            .map(GetOrdersResponseView::new)
            .map(RestWsResponseView::createGoodResponseView)
            .orElseThrow();
}
```

Как еще достать или выставить данные? REST API!

```
@Service
public class OrderService implements IOrderService {
...

public PaginatedElements<OrderModel> getOrders(
            Date modifiedAtStartingFrom, int pageNumber, int pageSize) {
    Page<OrderEntity> page =
            orderRepository.findOrdersWithPagination(
                    modifiedAtStartingFrom,//1
                    PageRequest.of(pageNumber, pageSize, Sort.by("id"))//2
            );
    return PaginatedElements.of(page, OrderModel::new);
}
```

Как еще достать или выставить данные? REST API!

```
public interface OrderRepository
       extends CrudRepository<OrderEntity, Long> {
...

default Page<OrderEntity> findOrdersWithPagination(
                Date modifiedAtStartingFrom, Pageable pageable) {
    if (isNull(modifiedAtStartingFrom)) {
        return findAll(pageable); //1
    }
    return findOrdersWithPaginationFilteredByModifiedAtStartingFrom(//2
                modifiedAtStartingFrom,
                pageable
    );
}
```

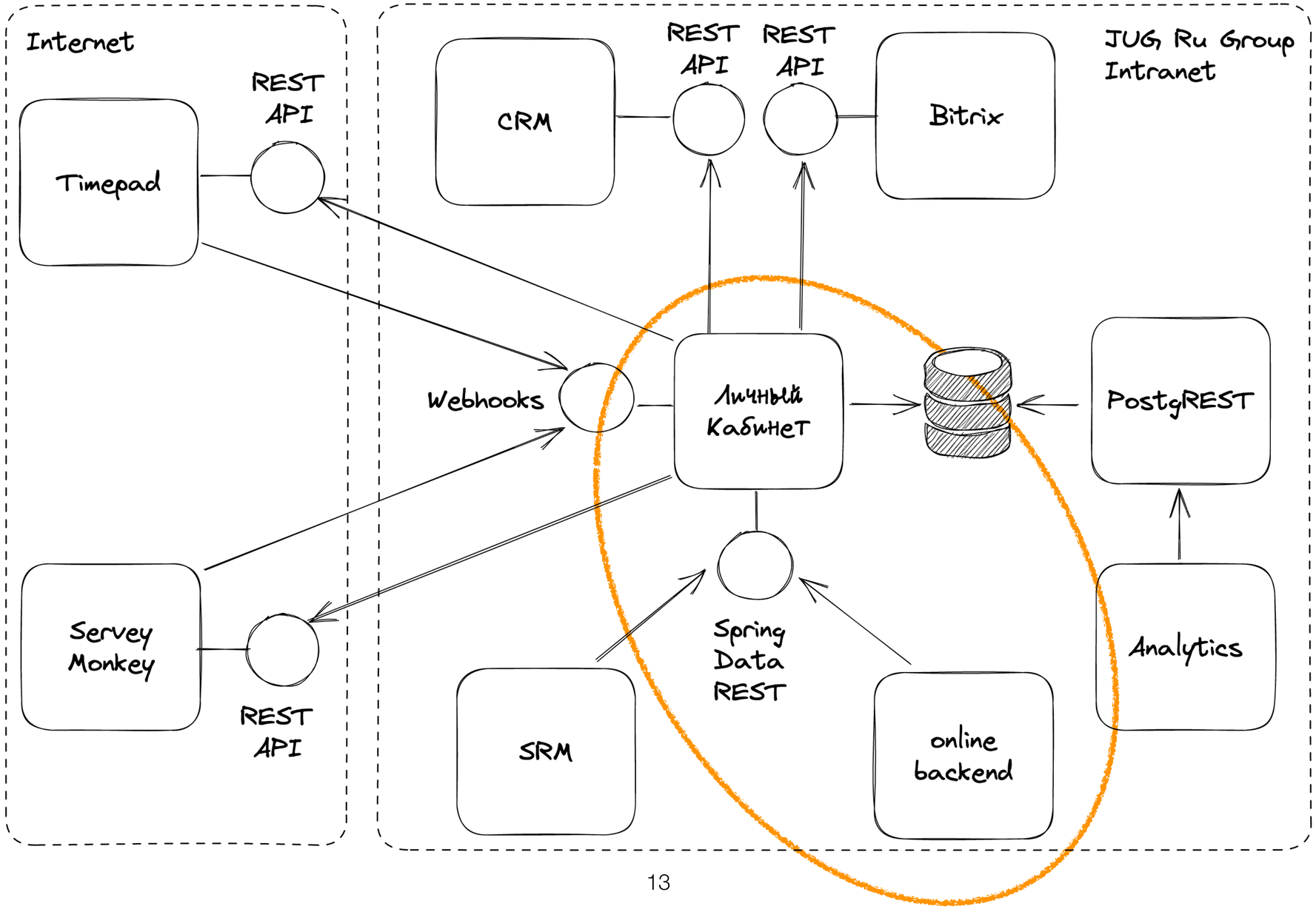
Как еще достать или выставить данные? REST API!

```
TimepadApiWebHook webHook = TimepadApiWebHook.builder()
        .organization(orgId)
        .status("ok") // -- 1
        .type("order_change") // -- 2
        .secret(timepadWebhooksSecretKey) // -- 3
        .url(timepadWebhookOnOrderChangedUrl) // -- 4
        .build();

timepadApiRestTemplate.postForObject( // -- 5
        “/v1/organizations/{orgId}/hooks", // -- 6
        webHook,
        TimepadApiWebHook.class,
        orgId);
```

Как еще достать или выставить данные? REST API!

```
@RequestMapping(value = "/restws/v1/webhooks/timepad/order", method = POST)
public @ResponseBody
RestWsResponseView processOrderWebHookFromTimepad(
        HttpServletRequest request,
        @RequestHeader("X-Hub-Signature") String signature,
        @RequestParam(required = false) boolean sync) throws IOException {
    String body = requireNonEmptyPlainRequestBody(request, "Timepad");
    if (sync) {
     webHookService.processTimepadOrderWebHookWithRetries(body, signature);
    } else {
     webHookService.processTimepadOrderWebHookAsync(body, signature);
    }
    return createGoodEmptyResponseView();
}
```

Кейз № 4.2


Internet
REST API
Timepad
Servey Monkey
REST API
JUG Ru Group Intranet
CRM
REST API
REST API
Bitrix
Webhooks
Личный Кабинет
PostgREST
Spring Data REST
SRM
online backend
Analytics

Как еще достать или выставить данные из БД? Spring Data REST!

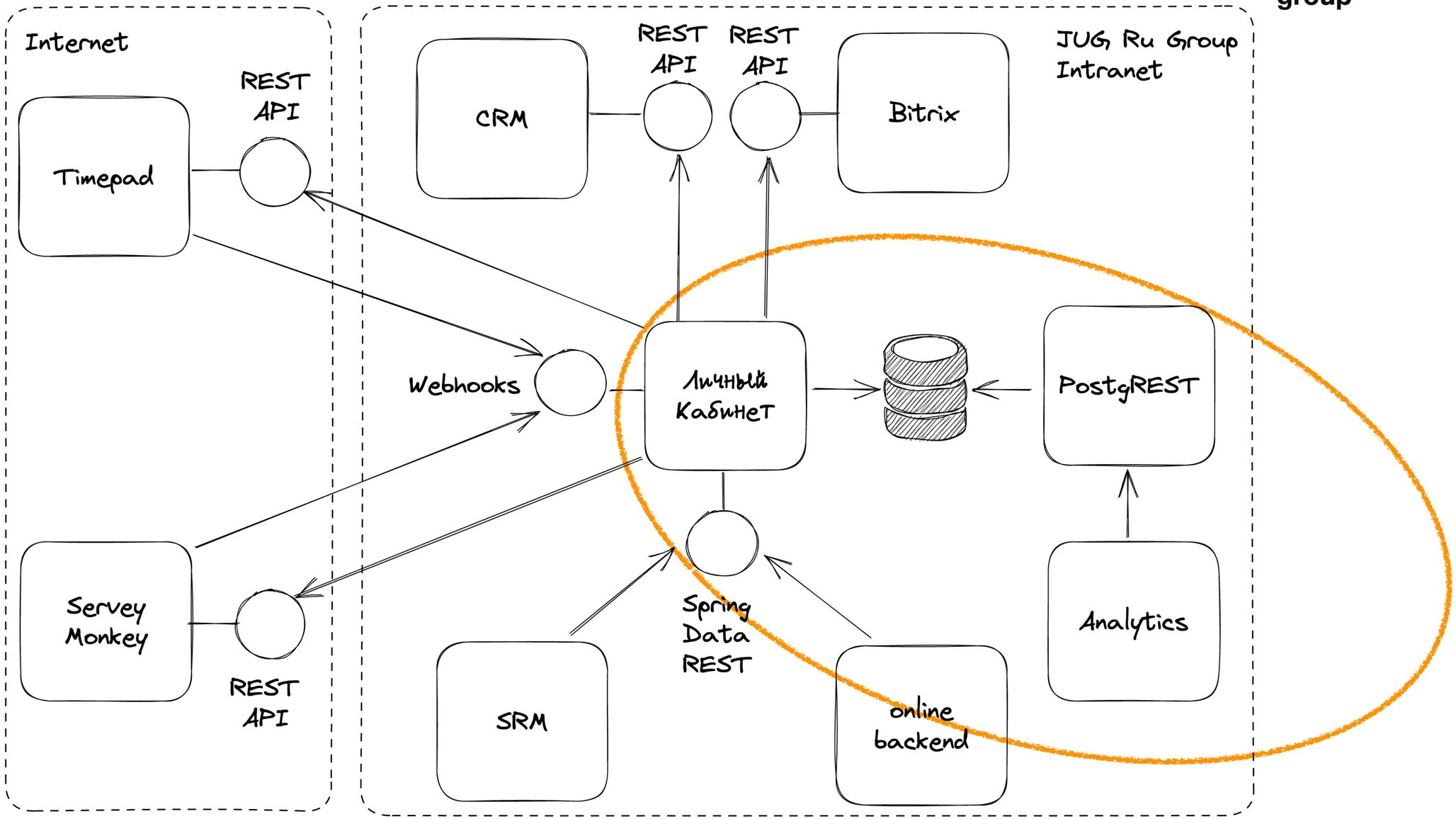
Как достать или выставить данные из БД? Spring Data REST!

```
spring.data.rest.base-path=/api/v1/internal/restdata          < — 1)
spring.data.rest.detection-strategy=annotated
spring.data.rest.default-page-size=20
spring.data.rest.max-page-size=100
...
@RepositoryRestResource                                       < — 2)
public interface TimepadEventRepository
  extends CrudRepository<TimepadEventEntity, TimepadEventEntity.IdClass> {

   TimepadEventEntity findById(long eventId);

   List<TimepadEventEntity> findByNameIgnoreCaseIsContaining(String text);

}
...
/api/v1/internal/restdata/timepadEventEntities{?page,size,sort} < — 3)
```

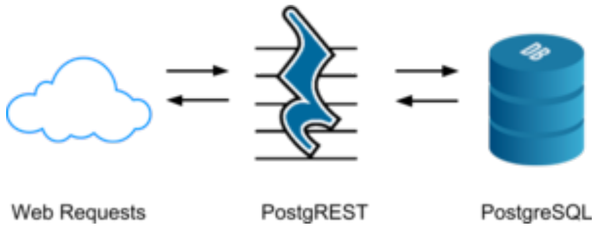
Как достать или выставить данные из БД? Spring Data REST!

```
@Configuration
@Order(2) < —— 4)
public class InternalSecurityConfig extends WebSecurityConfigurerAdapter {
@Override
protected void configure(HttpSecurity http) throws Exception {
    http.regexMatcher("^\\/api\\/v1\\/internal.*$")
            .csrf().disable()
            .formLogin().disable()
            .sessionManagement(sessionConfig -> sessionConfig
                    .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
            )
            .authorizeRequests(requestConfig -> requestConfig
                    .anyRequest().hasRole("INTERNAL_USER")
            )
            .httpBasic(basicConfig -> basicConfig
                    .authenticationEntryPoint(authenticationEntryPoint())
            );
}
```

Кейз № 4.3


Internet
REST API
Timepad
Servey Monkey
REST API
JUG Ru Group Intranet
CRM
REST API
REST API
Bitrix
Webhooks
Личный Кабинет
PostgREST
Spring Data REST
SRM
online backend
Analytics

Как еще достать или выставить данные из БД? PostgREST!



Как еще достать или выставить данные? PostgREST!

```
db-uri = "postgres://pg_user:pg_password@pg_host:5433/pg_db"
db-schemas = "datamart"
db-anon-role = "datamart_anon"

...

jwt-secret = "<PASSWORD MUST BE AT LEAST 32 CHARS LONG>”

...

GET /people?age=gte.18&student=is.true&order=age.desc&limit=15
```

https://postgrest.org/en/stable/

Чеклист клиента REST API

Чеклист клиента REST API

1. Периодическая джоба или событие

Чеклист клиента REST API

1. Периодическая джоба или событие

2. Постраничное зачитывание с фиксированной сортировкой и фильтром, например, modifiedAtStartingFrom

Чеклист клиента REST API

1. Периодическая джоба или событие

2. Постраничное зачитывание с фиксированной сортировкой и фильтром, например, modifiedAtStartingFrom

3. Вебхук для streaming, “но это не точно”

Чеклист клиента REST API

1. Периодическая джоба или событие
2. Постраничное зачитывание с фиксированной сортировкой и фильтром, например, modifiedAtStartingFrom
3. Вебхук для streaming, “но это не точно”
4. Обработка отказов, “дружелюбность”: Hystrix, Spring Retry, Failsafe

Чеклист клиента REST API

1. Периодическая джоба или событие
2. Постраничное зачитывание с фиксированной сортировкой и фильтром, например, modifiedAtStartingFrom
3. Вебхук для streaming, “но это не точно”
4. Обработка отказов, “дружелюбность”: Hystrix, Spring Retry, Failsafe
5. Обработка Rate Limits

REST API: Rate Limits

JUG.RU group

```
create table timepad_import_orders_loop_log (
    event_id bigint,                    -- < -- 1
    org_id bigint,                      -- < -- 2
    event_starts_at timestamp,          -- < -- 3
    event_finishes_at timestamp,        -- < -- 4
    start_from timestamp,               -- < -- 5
    last_success timestamp,             -- < -- 6
    total integer,                      -- < -- 7
    cursor integer,                     -- < -- 8
    status smallint,                    -- < -- 9
    created_date timestamp,             -- < -- 10
    modified_date timestamp             -- < -- 11
);
```

REST API: Rate Limits

JUG.RU group

```
timepadImportService.checkPreviousOrdersLoopCompletedAndPrepareForNextLoop(); //1
List<TimepadImportOrdersLoopLog> logs =
     timepadImportService.getCurrentOrdersLoopLog(); //2
logs.forEach(log -> { //3
  while (log.getStatus() != COMPLETED) { //4
    TimepadApiOrdersResponse response = retrieveOrdersFromTimepad(log); //5
    updateOrdersLoopLog(log, response); //6
    List<TimepadOrderEntity> orders = response.getValues();
    log.info("importing {} orders for org {} and event {}
         with total {}, next cursor {} and status {}", orders.size(),
         orgId, eventId, log.getTotal(), log.getCursor(), log.getStatus()
    );
    log = timepadImportService.importOrdersAndUpdateOrdersLoopLog(log, orders);//7
  }
});
```

Кластер из джобов: Shedlock

Кластер из джобов: Shedlock


```
@Configuration
@EnableScheduling
@EnableSchedulerLock(interceptMode = PROXY_METHOD,
defaultLockAtMostFor = "PT120M", defaultLockAtLeastFor = "PT10S")
public class BatchConfig implements SchedulingConfigurer {
    @Value("${mongo.db}")
    private String databaseName;
    @Bean
    public LockProvider lockProvider(MongoClient mongoClient) {
        return new MongoLockProvider(mongoClient.getDatabase(databaseName));
    }
    @Bean
    public LockingTaskExecutor lockingTaskExecutor(
              LockProvider lockProvider) {
        return new DefaultLockingTaskExecutor(lockProvider);
    }
}
```

Кластер из джобов: Shedlock


```
@Scheduled(cron = "${timepad.api.orders.cron.expression}")
@SchedulerLock(name = "TimepadBatchService.fetchAndImportOrders")
public void fetchAndImportOrders() {
    timepadOrderWorkerService.fetchAndImportOrders();
}
```

Кластер из джобов: Shedlock

JUG.RU group

postgresql

5 rows | Tx: Auto | DDL | CSV

```
WHERE name like 'Timepad%'
ORDER BY name
```

	name	lock_until	locked_at	locked_by
1	TimepadBatchService.fetchAndImportEvents	2023-03-05 14:07:10.001	2023-03-05 14:07:…	testing-lk-java-backend-5f8f6957b4-w2wwx
2	TimepadBatchService.fetchAndImportInvoices	2023-03-05 17:12:51.208	2023-03-05 17:12:…	testing-lk-java-backend-5f8f6957b4-w2wwx
3	TimepadBatchService.fetchAndImportOrders	2023-03-05 17:43:03.504	2023-03-05 17:42:…	testing-lk-java-backend-5f8f6957b4-rngsl
4	TimepadBatchService.fetchAndImportOrganisations	2023-03-05 06:06:10.001	2023-03-05 06:06:…	testing-lk-java-backend-5f8f6957b4-rngsl
5	TimepadBatchService.updateOrCreateWebHooksOnOrderChange	2023-03-05 05:32:10.002	2023-03-05 05:32:…	testing-lk-java-backend-5f8f6957b4-rngsl

mongodb

	_id	lockUntil	lockedAt	lockedBy
1	ContentfulEventsCalendarService.preload	2023-02-17T15:55:16.480Z	2023-02-17T15:55:15.480Z	testing-online-mvp-backer
2	ContentfulImportBatchService.importEventP…	2023-03-05T17:57:03.355Z	2023-03-05T17:57:00.000Z	testing-online-mvp-backer
3	ContentfulImportBatchService.importLegacy…	2023-02-28T14:41:15.482Z	2023-02-28T14:40:37.905Z	testing-online-mvp-backer
4	ContentfulImportBatchService.importSchedu…	2023-03-05T17:54:53.072Z	2023-03-05T17:54:00.001Z	testing-online-mvp-backer

Кластер из джобов: Shedlock


```
lockingTaskExecutor.executeWithLock(
        (Runnable) this::PreloadFromStoreIfNeeded,
        new LockConfiguration(
            this.getClass().getSimpleName() + “.preload",
            Duration.ofMinutes(3),
            Duration.ofSeconds(1)
        )
);
```

k8s: CronJob

CronJob

FEATURE STATE: `Kubernetes v1.21 [stable]`

A *CronJob* creates Jobs on a repeating schedule.

CronJob is meant for performing regular scheduled actions such as backups, report generation, and so on. One CronJob object is like one line of a *crontab* (cron table) file on a Unix system. It runs a job periodically on a given schedule, written in Cron format.

CronJobs have limitations and idiosyncrasies. For example, in certain circumstances, a single CronJob can create multiple concurrent Jobs. See the limitations below.

https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/

k8s: CronJob

CronJob

FEATURE STATE: `Kubernetes v1.21 [stable]`

A *CronJob* creates Jobs on a repeating schedule.

CronJob is meant for performing regular scheduled actions such as backups, report generation, and so on. One CronJob object is like one line of a *crontab* (cron table) file on a Unix system. It runs a job periodically on a given schedule, written in Cron format.

CronJobs have limitations and idiosyncrasies. For example, in certain circumstances, a single CronJob can create multiple concurrent Jobs. See the limitations below.

https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/

k8s: CronJob

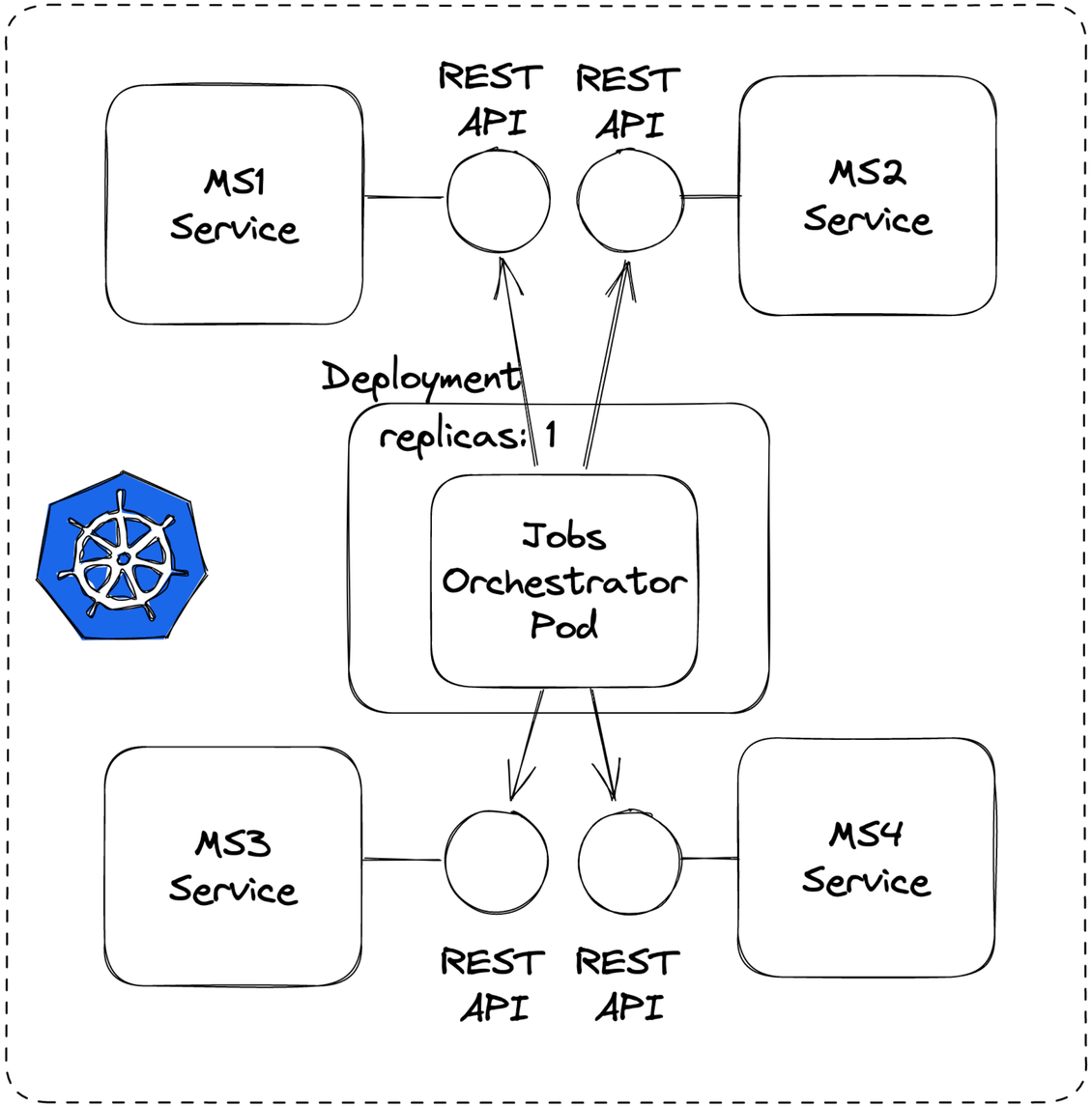
Job creation

A CronJob creates a Job object approximately once per execution time of its schedule. The scheduling is approximate because there are certain circumstances where two Jobs might be created, or no Job might be created. Kubernetes tries to avoid those situations, but do not completely prevent them. Therefore, the Jobs that you define should be *idempotent*.

If `startingDeadlineSeconds` is set to a large value or left unset (the default) and if `concurrencyPolicy` is set to `Allow`, the jobs will always run at least once.

https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/#cron-job-limitations

Кейз №5


REST API
REST API
MS1 Service
MS2 Service
Deployment
replicas: 1
Jobs Orchestrator Pod
MS3 Service
MS4 Service
REST API
REST API

JUG.RU group

Аудит изменений: Hibernate Envers

Аудит изменений: Hibernate Envers


```
spring.jpa.properties.org.hibernate.envers.audit_table_prefix=
spring.jpa.properties.org.hibernate.envers.audit_table_suffix=_audit
spring.jpa.properties.org.hibernate.envers.revision_field_name=revision
spring.jpa.properties.org.hibernate.envers.revision_type_field_name=revision_type

...

create table revision_info
(
    id        integer      not null  -- 1
        primary key,
    timestamp bigint       not null, -- 2
    username  varchar(255) not null  -- 3
);
```

Аудит изменений: Hibernate Envers


```
create table vip_user
(
    id            bigint       not null
        constraint pk_vip_user primary key,
    lang          varchar(255),
    password_hash varchar(255),
    status        int2         not null default 0,
    comment       varchar(255),
    created_date  timestamp    not null default now(),
    modified_date timestamp    not null default now()
);
```

Аудит изменений: Hibernate Envers


```
create table vip_user_audit
(
    id            int8          not null,
    lang          varchar(255),
    password_hash varchar(255),
    status        int2          not null default 0,
    comment       varchar(255),
    revision      int4          not null,
    revision_type int2          not null,
    primary key (id, revision)
);
```

Аудит изменений: Hibernate Envers


```
create table vip_user_audit
(
    id            int8          not null,
    lang          varchar(255),
    password_hash varchar(255),
    status        int2          not null default 0,
    comment       varchar(255),
    revision      int4          not null,
    revision_type int2          not null,
    primary key (id, revision)
);
```

Аудит изменений: Hibernate Envers


```
@Entity
@Table(name = "VIP_USER")
@EntityListeners(AuditingEntityListener.class)
@Audited
public class VipUserEntity {
    @Id
    private long id;
    private String passwordHash;
    private String comment;
    private UserStatus status;
    private String lang;
    @NotAudited
    @CreatedDate
    private Date createdDate;
    @NotAudited
    @LastModifiedDate
    private Date modifiedDate;
```

Аудит изменений: Hibernate Envers


```
@Entity
@RevisionEntity(UsernameRevisionListener.class)
@Table(name = "REVISION_INFO")
public class RevisionInfoEntity extends DefaultRevisionEntity {
    private String username;
}
...
public class UsernameRevisionListener implements RevisionListener {
    @Autowired
    private IAuthenticationFacade authenticationFacade;
    @Override
    public void newRevision(Object revisionEntity) {
        RevisionInfoEntity revision = (RevisionInfoEntity) revisionEntity;
        revision.setUsername(
            authenticationFacade.getAuthenticatedUserName().orElse("anonymous"));
    }
}
```

Выводы (часть 2)

Выводы (часть 2)

1. Подход к интеграции по данным через REST API

Выводы (часть 2)

1. Подход к интеграции по данным через REST API

2. Постраничное чтение, фильтр, сортировка и вебхуки для стриминга

Выводы (часть 2)

1. Подход к интеграции по данным через REST API

2. Постраничное чтение, фильтр, сортировка и вебхуки для стриминга

3. Оптимизация времени на интеграцию со Spring Data REST и postgrest

Выводы (часть 2)

1. Подход к интеграции по данным через REST API

2. Постраничное чтение, фильтр, сортировка и вебхуки для стриминга

3. Оптимизация времени на интеграцию со Spring Data REST и postgrest

4. Контрольная таблица как средство работы с Rate Limits

Выводы (часть 2)

1. Подход к интеграции по данным через REST API

2. Постраничное чтение, фильтр, сортировка и вебхуки для стриминга

3. Оптимизация времени на интеграцию со Spring Data REST и postgrest

4. Контрольная таблица как средство работы с Rate Limits

5. ShedLock

Выводы (часть 2)

1. Подход к интеграции по данным через REST API

2. Постраничное чтение, фильтр, сортировка и вебхуки для стриминга

3. Оптимизация времени на интеграцию со Spring Data REST и postgrest

4. Контрольная таблица как средство работы с Rate Limits

5. ShedLock

6. K8S: CronJob, паттерн “оркестратор джобов”

Выводы (часть 2)

1. Подход к интеграции по данным через REST API

2. Постраничное чтение, фильтр, сортировка и вебхуки для стриминга

3. Оптимизация времени на интеграцию со Spring Data REST и postgrest

4. Контрольная таблица как средство работы с Rate Limits

5. ShedLock

6. K8S: CronJob, паттерн “оркестратор джобов”

7. Аудит данных на Hibernate Envers

Спасибо!

@vlakra
vladimir.krasilschik@gmail.com

План доклада (часть 2)

План доклада (часть 2)

1. Обсуждаем подход к интеграции по данным через REST API

План доклада (часть 2)

1. Обсуждаем подход к интеграции по данным через REST API

2. Детализируем постраничное чтение, фильтр, сортировка и вебхуки для стриминга

План доклада (часть 2)

1. Обсуждаем подход к интеграции по данным через REST API

2. Детализируем постраничное чтение, фильтр, сортировка и вебхуки для стриминга

3. Представляем контрольную таблицу как средство борьбы с Rate Limits

План доклада (часть 2)

1. Обсуждаем подход к интеграции по данным через REST API

2. Детализируем постраничное чтение, фильтр, сортировка и вебхуки для стриминга

3. Представляем контрольную таблицу как средство борьбы с Rate Limits

4. Оптимизируем времени на интеграцию со Spring Data REST и postgrest

План доклада (часть 2)

1. Обсуждаем подход к интеграции по данным через REST API

2. Детализируем постраничное чтение, фильтр, сортировка и вебхуки для стриминга

3. Представляем контрольную таблицу как средство борьбы с Rate Limits

4. Оптимизируем времени на интеграцию со Spring Data REST и postgrest

5. Рассматриваем ShedLock

План доклада (часть 2)

1. Обсуждаем подход к интеграции по данным через REST API

2. Детализируем постраничное чтение, фильтр, сортировка и вебхуки для стриминга

3. Представляем контрольную таблицу как средство борьбы с Rate Limits

4. Оптимизируем времени на интеграцию со Spring Data REST и postgrest

5. Рассматриваем ShedLock

6. Критикуем K8S CronJob, предлагаем паттерн “оркестратор джобов”

План доклада (часть 2)

1. Обсуждаем подход к интеграции по данным через REST API

2. Детализируем постраничное чтение, фильтр, сортировка и вебхуки для стриминга

3. Представляем контрольную таблицу как средство борьбы с Rate Limits

4. Оптимизируем времени на интеграцию со Spring Data REST и postgrest

5. Рассматриваем ShedLock

6. Критикуем K8S CronJob, предлагаем паттерн “оркестратор джобов”

7. Добиваем аудитом данных на Hibernate Envers