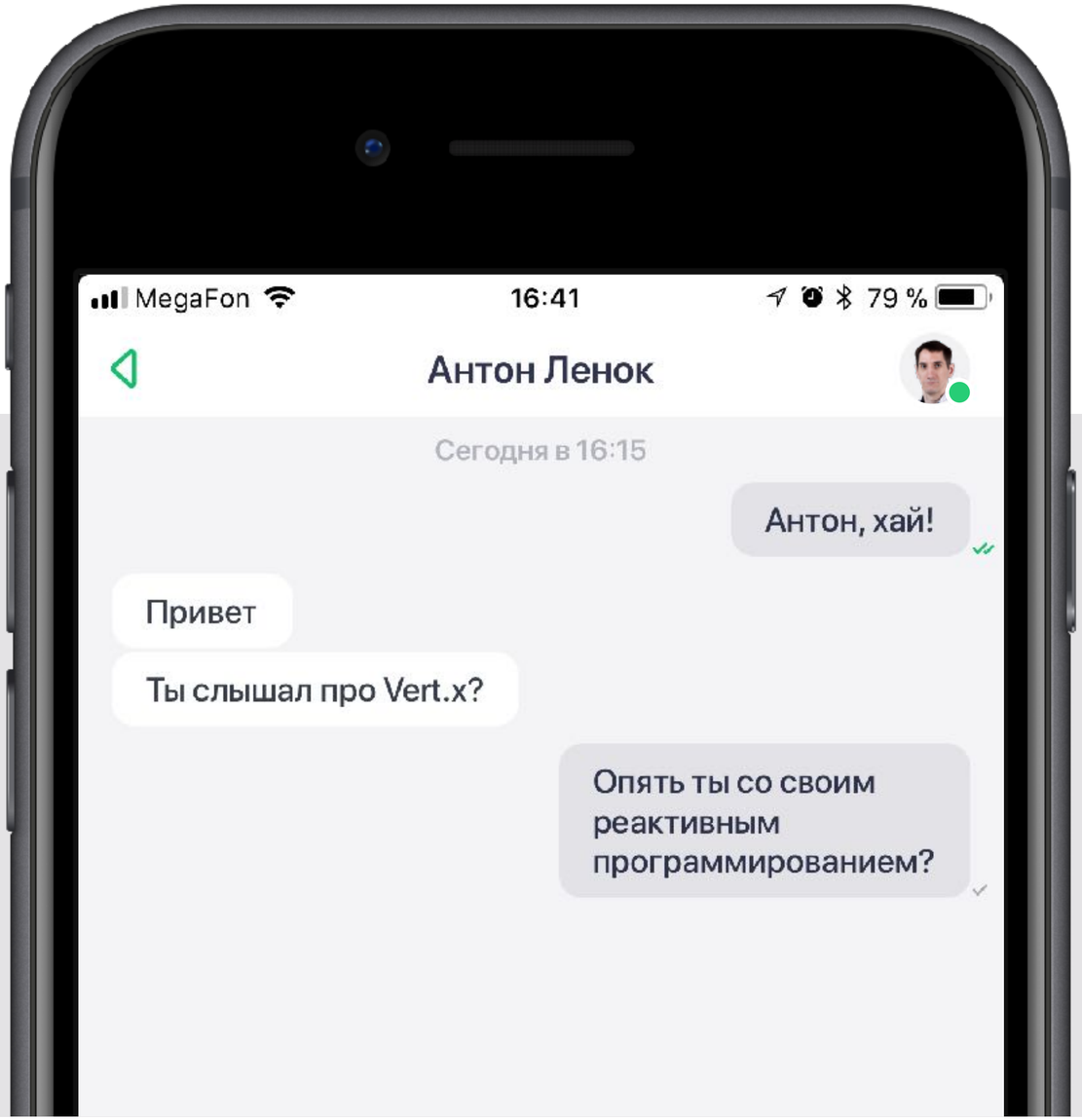
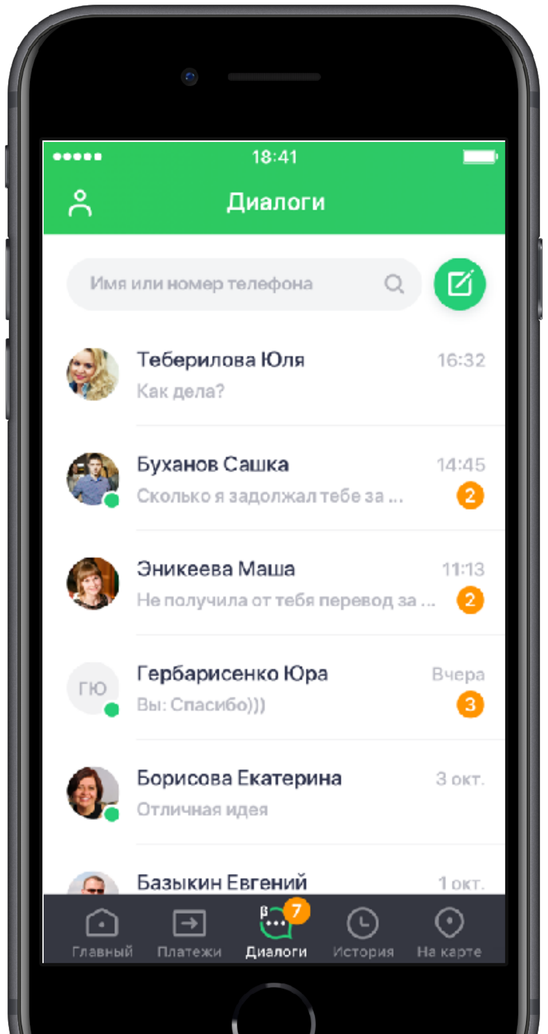
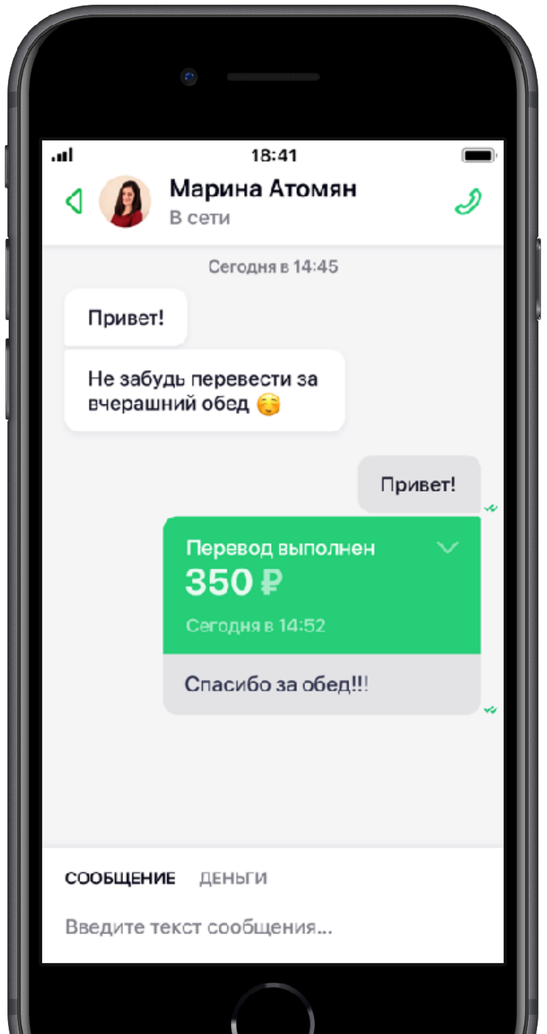
Сбербанк Онлайн

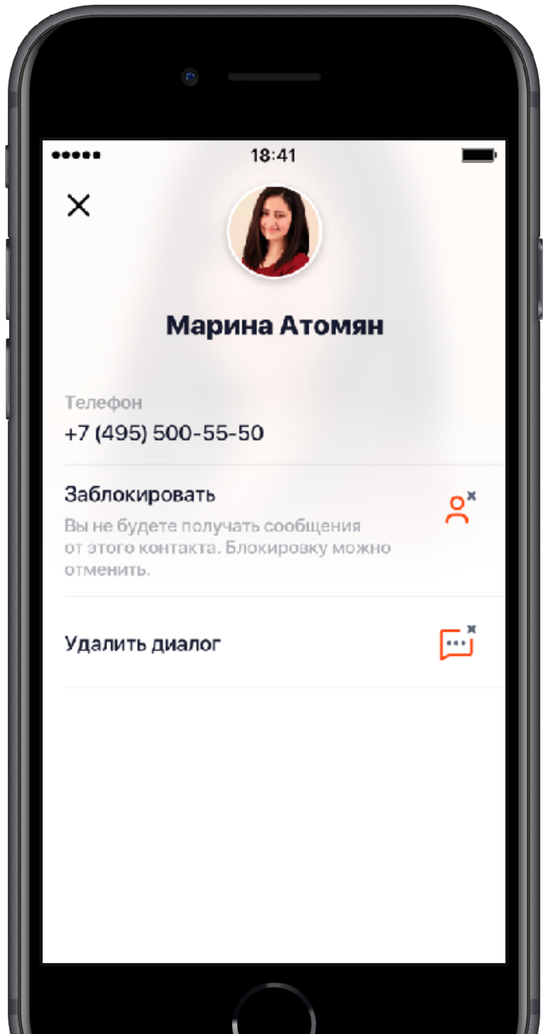
Реактивное программирование на Vert.x

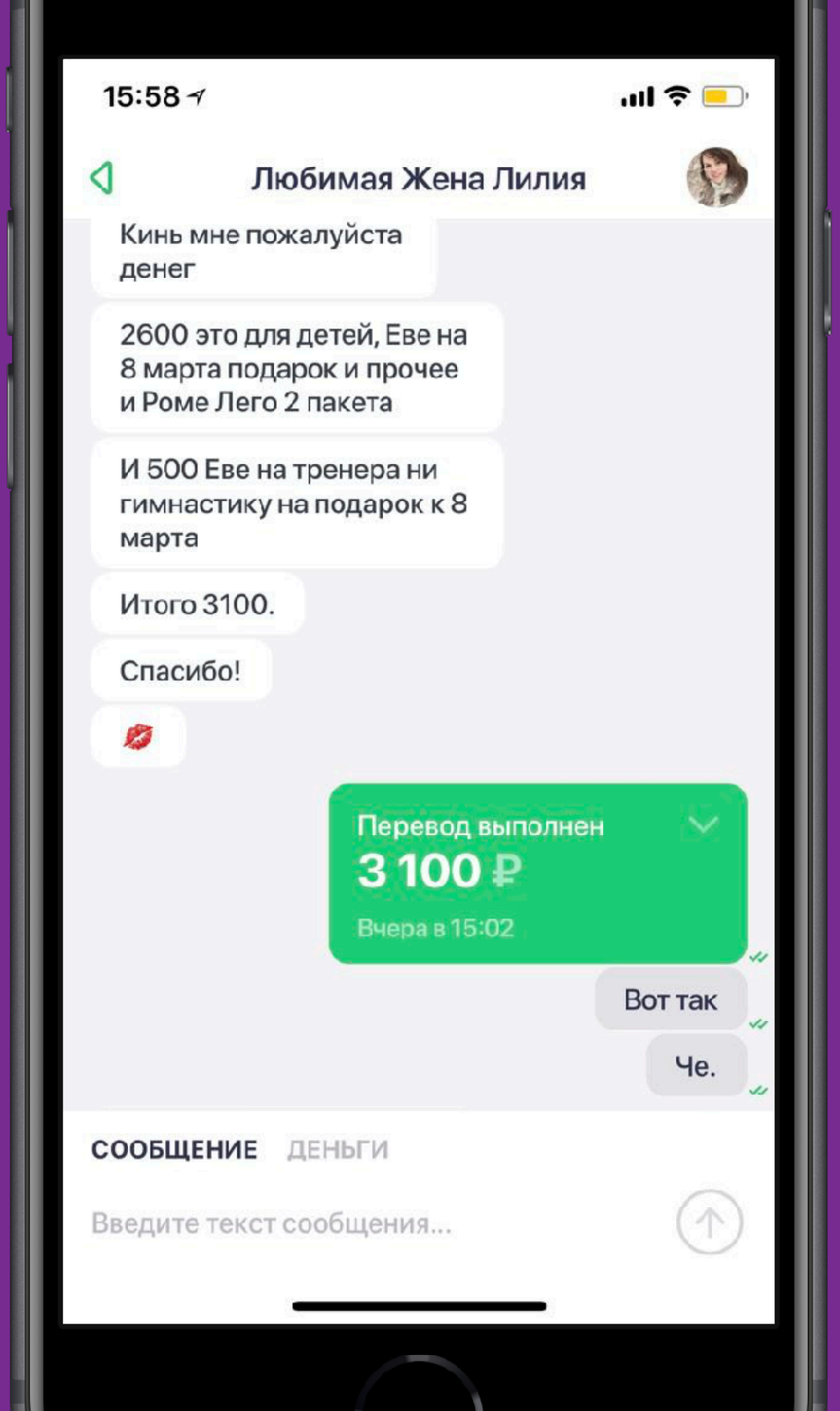


Диалоги в Сбербанк Онлайн








15:58
Любимая Жена Лилия
Кинь мне пожалуйста денег
2600 это для детей, Еве на 8 марта подарок и прочее и Роме Лего 2 пакета
И 500 Еве на тренера ни гимнастику на подарок к 8 марта
Итого 3100.
Спасибо!
Перевод выполнен
3 100 ₽
Вчера в 15:02
Вот так
Че.
СООБЩЕНИЕ
ДЕНЬГИ
Введите текст сообщения...

Сбербанк Онлайн

Ленок Антон

Серверная разработка

Проект: Диалоги
в Сбербанк Онлайн

AILenok.SBT@sberbank.ru

Задача

Прием и отправка сообщений

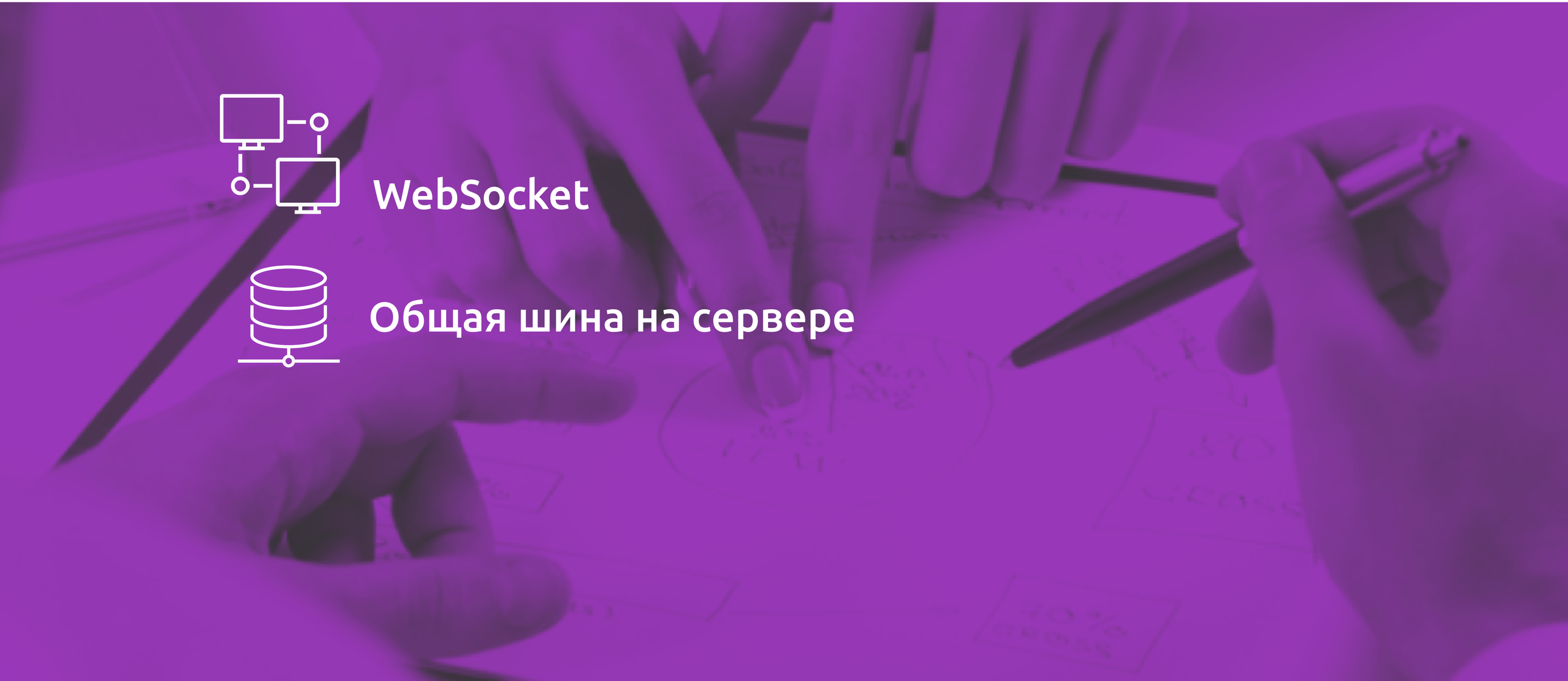
>1 Скорость отправки > приёма сообщения <1 секунды

В начале 10 человек онлайн

Целевое 100 000 человек онлайн

Быстрое добавление функций

Решение



WebSocket

Общая шина на сервере

Vert.x описание

- Создатель Тим Фокс
- Встроенная общая шина
- Абстракция над Java Concurrency
- Реактивный

We Are Reactive

Полиглот
для языков JVM

- Java
- JavaScript
- Ruby
- Groovy
- Scala
- Kotlin

На чем фокусируемся

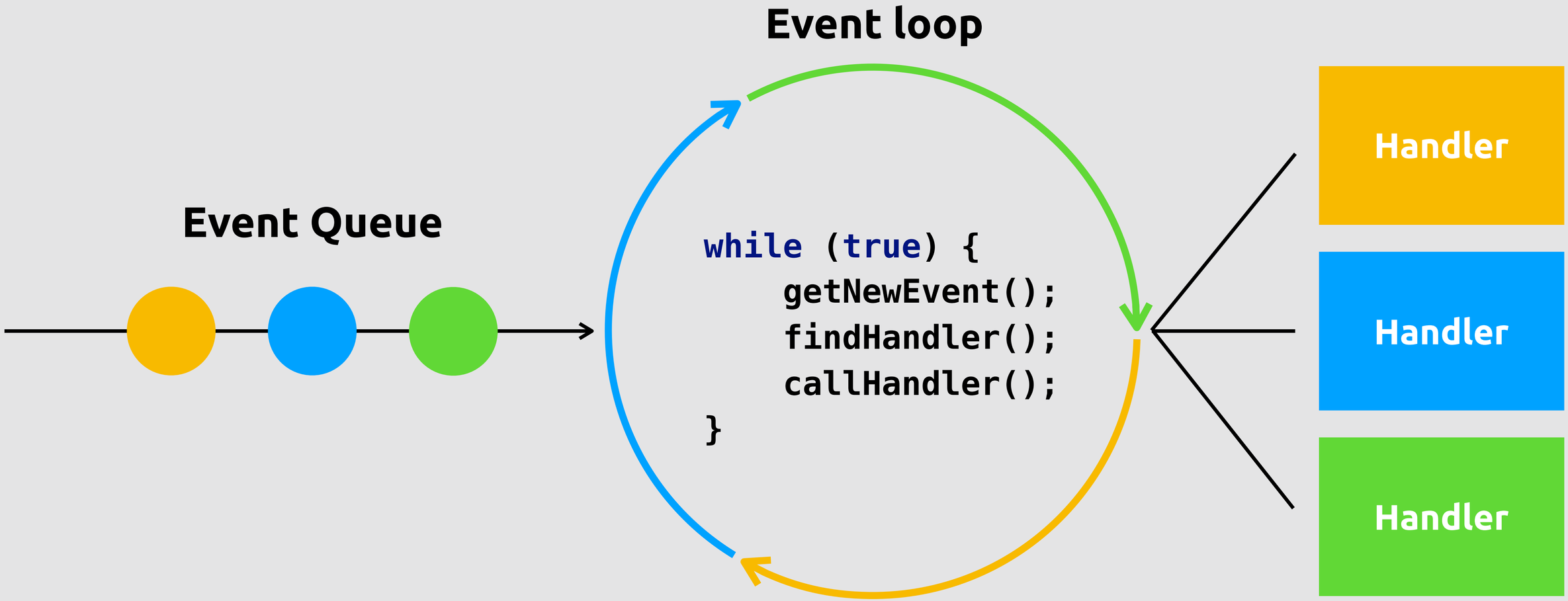
Подробно:

- Принципы реактивного программирования
- Vert.x

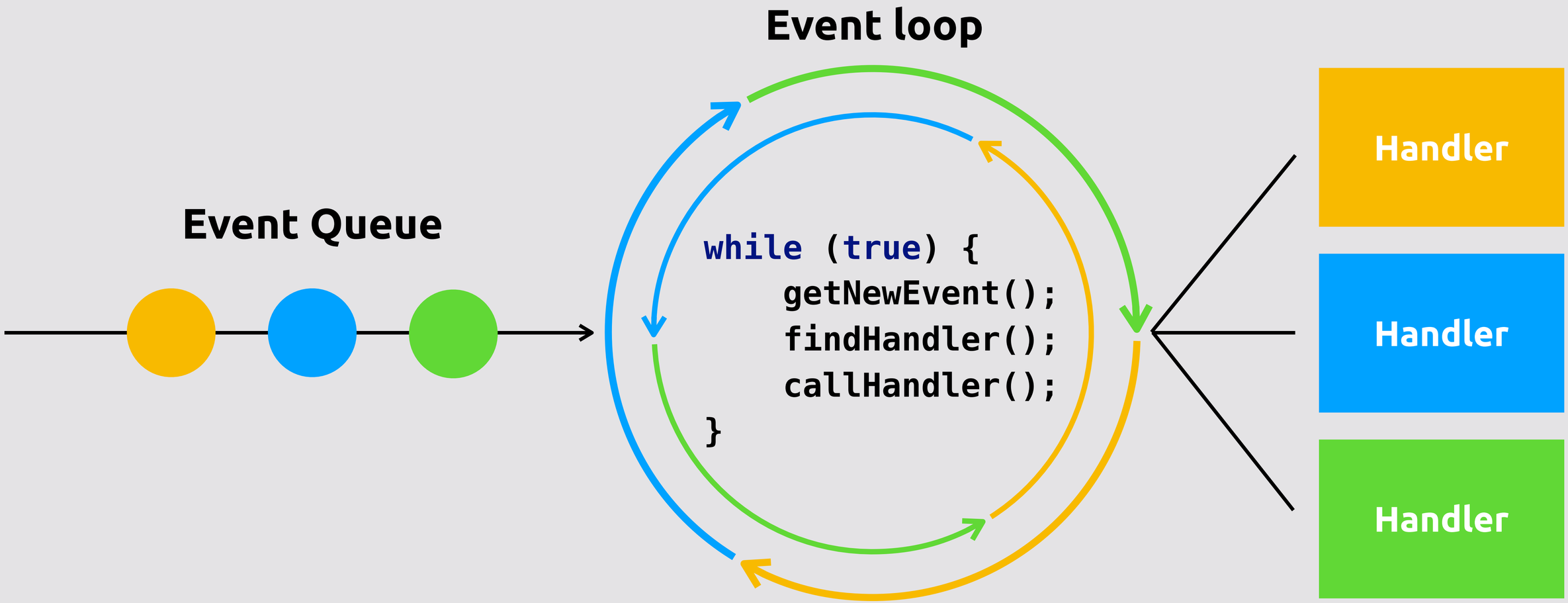
Не концентрируемся:

- Kubernetes
- Mongo DB

Реактивное программирование



Multi-Reactor



Первое приложение


```
import io.vertx.core.AbstractVerticle;
import io.vertx.core.eventbus.Message;

public class HelloVerticle extends AbstractVerticle {

    @Override
    public void start() {

        vertx.eventBus().consumer("greetings", this::hello);

        vertx.eventBus().publish("greetings", "My first message");
    }

    private void hello(Message<String> data) {
        String message = data.body();
        System.out.println(message);
    }
}
```

Первое приложение


```
import io.vertx.core.AbstractVerticle;
import io.vertx.core.eventbus.Message;

public class HelloVerticle extends AbstractVerticle {

    @Override
    public void start() {

        vertx.eventBus().consumer("greetings", this::hello);

        vertx.eventBus().publish("greetings", "My first message");
    }

    private void hello(Message<String> data) {
        String message = data.body();
        System.out.println(message);
    }
}
```

Первое приложение


```
import io.vertx.core.AbstractVerticle;
import io.vertx.core.eventbus.Message;

public class HelloVerticle extends AbstractVerticle {

    @Override
    public void start() {

        vertx.eventBus().consumer("greetings", this::hello);

        vertx.eventBus().publish("greetings", "My first message");
    }

    private void hello(Message<String> data) {
        String message = data.body();
        System.out.println(message);
    }
}
```

Первое приложение


```
import io.vertx.core.AbstractVerticle;
import io.vertx.core.eventbus.Message;

public class HelloVerticle extends AbstractVerticle {

    @Override
    public void start() {

        vertx.eventBus().consumer("greetings", this::hello);

        vertx.eventBus().publish("greetings", "My first message");
    }

    private void hello(Message<String> data) {
        String message = data.body();
        System.out.println(message);
    }
}
```

Первое приложение


```
import io.vertx.core.AbstractVerticle;
import io.vertx.core.eventbus.Message;

public class HelloVerticle extends AbstractVerticle {

    @Override
    public void start() {

        vertx.eventBus().consumer("greetings", this::hello);

        vertx.eventBus().publish("greetings", "My first message");
    }

    private void hello(Message<String> data) {
        String message = data.body();
        System.out.println(message);
    }
}
```

Первое приложение


```
import io.vertx.core.AbstractVerticle;
import io.vertx.core.eventbus.Message;

public class HelloVerticle extends AbstractVerticle {

    @Override
    public void start() {

        vertx.eventBus().consumer("greetings", this::hello);

        vertx.eventBus().publish("greetings", "My first message");
    }

    private void hello(Message<String> data) {
        String message = data.body();
        System.out.println(message);
    }
}
```

Первое приложение


```
import io.vertx.core.AbstractVerticle;
import io.vertx.core.eventbus.Message;

public class HelloVerticle extends AbstractVerticle {

    @Override
    public void start() {

        vertx.eventBus().consumer("greetings", this::hello);

        vertx.eventBus().publish("greetings", "My first message");
    }

    private void hello(Message<String> data) {
        String message = data.body();
        System.out.println(message);
    }
}
```

Первое приложение


```
import io.vertx.core.Vertx;
import org.example.verticle.HelloVerticle;

public class Starter {

    public static void main(String[] args) {
        Vertx.vertx().deployVerticle(new HelloVerticle());
    }
}
```

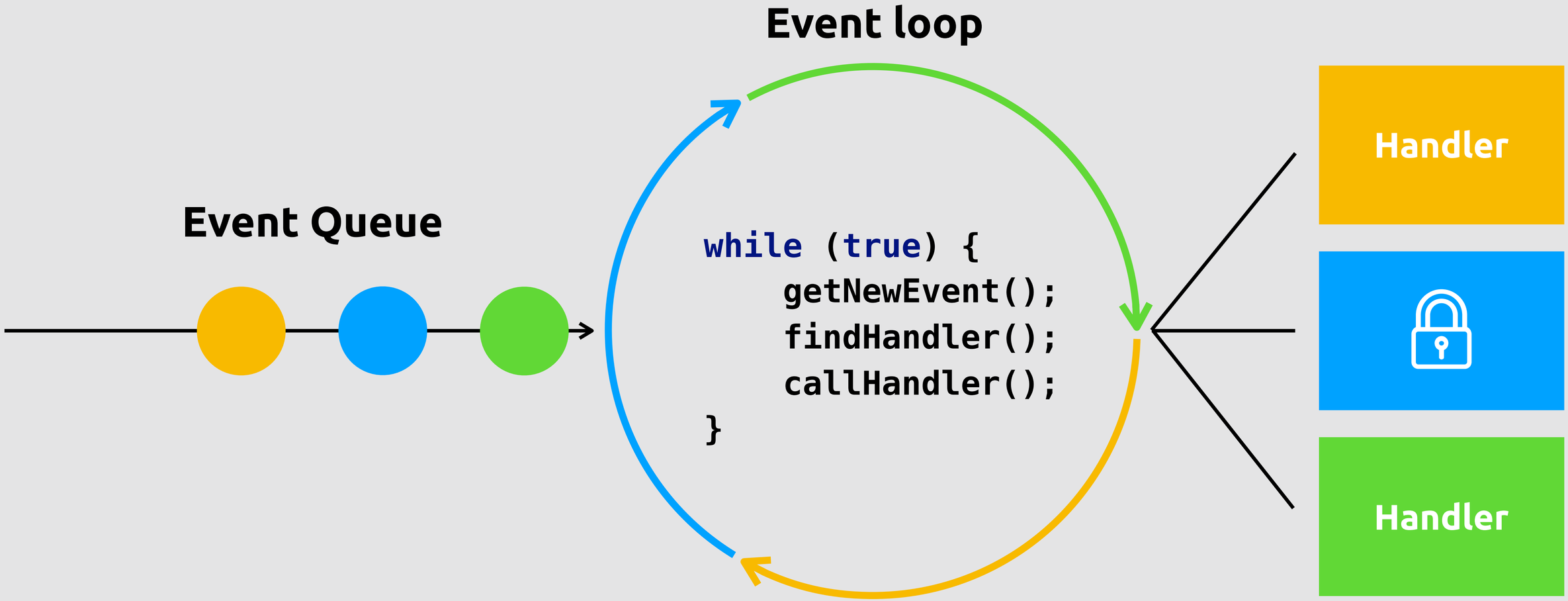
Первое приложение


```
import io.vertx.core.Vertx;
import org.example.verticle.HelloVerticle;

public class Starter {

    public static void main(String[] args) {
        Vertx.vertx().deployVerticle(new HelloVerticle());
    }
}
```

Научились

- Первое приложение

Блокирующие операции



Блокирующие операции


```
vertx.eventBus().consumer("block", this::blockingCode);


void blockingCode(Message<String> m) {
    for (long i = 0; i < Long.MAX_VALUE; i++) ;
}
```

Блокирующие операции


```
vertx.eventBus().consumer("block", this::blockingCode);


void blockingCode(Message<String> m) {
    vertx.executeBlocking(future -> {
        for (long i = 0; i < Long.MAX_VALUE; i++) ;
        future.complete("ok");
    }, res -> System.out.println(res.result()));
}
```

Блокирующие операции


```
vertx.eventBus().consumer("block", this::blockingCode);


void blockingCode(Message<String> m) {
    vertx.executeBlocking(future -> {
        for (long i = 0; i < Long.MAX_VALUE; i++) ;
        future.complete("ok");
    }, res -> System.out.println(res.result()));
}
```

Научились

- Первое приложение
- Блокирующие операции

Протокол

```
{
   "text"    : "Hello",
   "address": "nickname"
}
```

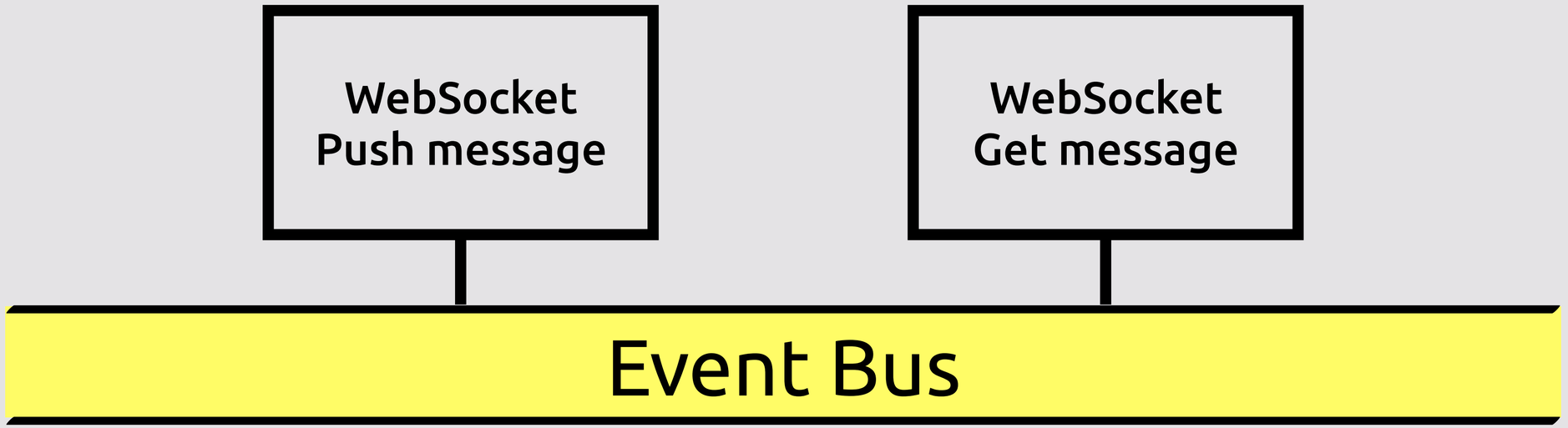
Клиент

```
var socket =
new WebSocket(
'ws://HOST/token/ADDRESS'
)
```

Клиент

```
socket.send(
'{"text"    :"Hello",
  "address":"receiver"}'
)
```

Общая шина v1.0



Отправка сообщений


```
public class WsServerVerticle extends
AbstractVerticle {
 @Override
 public void start() {
  vertx.createHttpServer()
  .websocketHandler(this::createWebSocketServer)
  .listen(8080);
 }
}
```

Отправка сообщений


```
@Override
public void start() {
  vertx.createHttpServer()
  .websocketHandler(this::createWebSocketServer)
  .listen(8080);
}
```

Отправка сообщений


```
@Override
public void start() {
   vertx.createHttpServer()
   .websocketHandler(this::createWebSocketServer)
   .listen(8080);
}
```

Отправка сообщений


```
@Override
public void start() {
  vertx.createHttpServer()
  .websocketHandler(this::createWebSocketServer)
  .listen(8080);
}
```

Отправка сообщений


```
@Override
public void start() {
   vertx.createHttpServer()
   .websocketHandler(this::createWebSocketServer)
   .listen(8080);
}
void createWebSocketServer
   (ServerWebSocket wsServer) {
}
```

Отправка сообщений


```
void createWebSocketServer(ServerWebSocket wsServer) {
 vertx.eventBus().<String>consumer(wsServer.path(), data -> {
  wsServer.writeFinalTextFrame(data.body());
  data.reply("ok");
 });
}
```

Отправка сообщений


```
void createWebSocketServer(ServerWebSocket wsServer) {
 vertx.eventBus().<String>consumer(wsServer.path(), data -> {
  wsServer.writeFinalTextFrame(data.body());
  data.reply("ok");
 });
}
```

Отправка сообщений


```
var socket =
new WebSocket(
'ws://HOST/token/ADDRESS'
)
```

Отправка сообщений


```
void createWebSocketServer(ServerWebSocket wsServer) {
 vertx.eventBus().<String>consumer(wsServer.path(), data -> {
  wsServer.writeFinalTextFrame(data.body());
  data.reply("ok");
 });
}
```

Отправка сообщений


```
void createWebSocketServer(ServerWebSocket wsServer) {
 vertx.eventBus().<String>consumer(wsServer.path(), data -> {
  wsServer.writeFinalTextFrame(data.body());
  data.reply("ok");
 });
}
```

Отправка сообщений


```
void createWebSocketServer(ServerWebSocket wsServer) {
 vertx.eventBus().<String>consumer(wsServer.path(), data -> {
  wsServer.writeFinalTextFrame(data.body());
  data.reply("ok");
 });
}
```

Прием сообщений


```
@Override
public void start() {
  vertx.createHttpServer()
  .websocketHandler(this::createWebSocketServer)
  .listen(8080);
}

void createWebSocketServer
  (ServerWebSocket wsServer) {
}
```

Прием сообщений


```
void createWebSocketServer(ServerWebSocket wsServer) {
 wsServer.frameHandler(wsFrame -> {
  Data data = Json.decodeValue(wsFrame.textData(), Data.class);
  String token = "/token/" + data.getAddress();
  vertx.eventBus().publish(
   token, wsFrame.textData());
 });
}
```

Прием сообщений


```
void createWebSocketServer(ServerWebSocket wsServer) {
 wsServer.frameHandler(wsFrame -> {
  Data data = Json.decodeValue(wsFrame.textData(), Data.class);
  String token = "/token/" + data.getAddress();
  vertx.eventBus().publish(
   token, wsFrame.textData());
 });
}
```

Прием сообщений


```
void createWebSocketServer(ServerWebSocket wsServer) {
 wsServer.frameHandler(wsFrame -> {
  Data data = Json.decodeValue(wsFrame.textData(), Data.class);
  String token = "/token/" + data.getAddress();
  vertx.eventBus().publish(
   token, wsFrame.textData());
 });
}
```

Прием сообщений


```
{
   "text"    : "Hello",
   "address": "nickname"
}
```

Прием сообщений


```
void createWebSocketServer(ServerWebSocket wsServer) {
 wsServer.frameHandler(wsFrame -> {
  Data data = Json.decodeValue(wsFrame.textData(), Data.class);
  String token = "/token/" + data.getAddress();
  vertx.eventBus().publish(
   token, wsFrame.textData());
 });
}
```

Прием сообщений


```
void createWebSocketServer(ServerWebSocket wsServer) {
 wsServer.frameHandler(wsFrame -> {
  Data data = Json.decodeValue(wsFrame.textData(), Data.class);
  String token = "/token/" + data.getAddress();
  vertx.eventBus().publish(
   token, wsFrame.textData());
 });
}
```

Прием сообщений


```
void createWebSocketServer(ServerWebSocket wsServer) {
 vertx.eventBus().<String>consumer(wsServer.path(), data -> {
  wsServer.writeFinalTextFrame(data.body());
  data.reply("ok");
 });
}
```

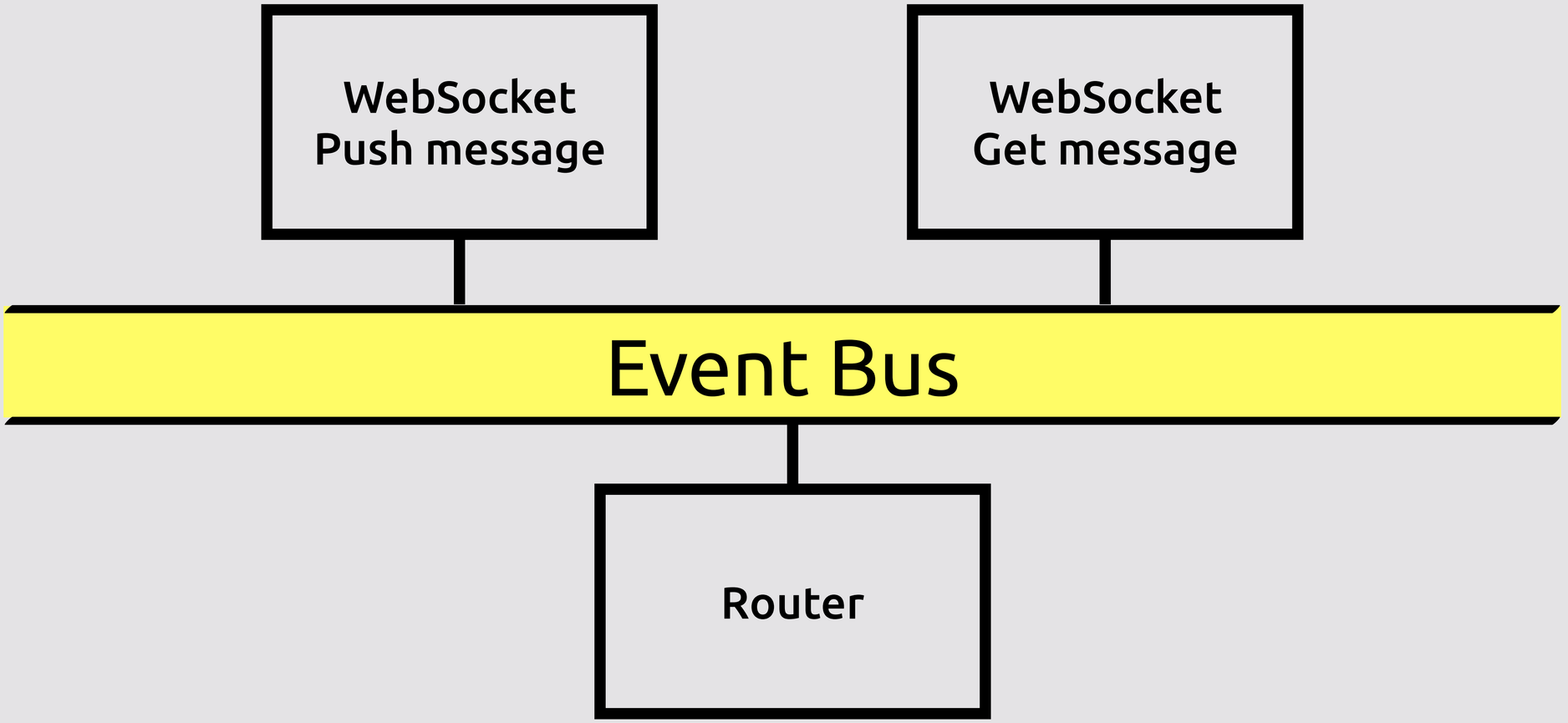
Прием сообщений

Demo

Научились

- Первое приложение
- Блокирующие операции
- WebSocket

Общая шина v2.0



Маршрутизатор


```
void createWebSocketServer(ServerWebSocket wsServer) {
 wsServer.frameHandler(wsFrame -> {
  Data data = Json.decodeValue(wsFrame.textData(), Data.class);
  String token = "/token/" + data.getAddress();
  vertx.eventBus().publish(
  token, wsFrame.textData());
 });
}
```

Маршрутизатор


```
void createWebSocketServer(ServerWebSocket wsServer) {
 wsServer.frameHandler(wsFrame -> {


   vertx.eventBus().publish(
   "router", wsFrame.textData());
 });
}
```

Маршрутизатор


```
public class RouterVerticle extends AbstractVerticle {
 @Override
 public void start() {
  vertx.eventBus().consumer("router", this::router);
 }

 void router(Message<String> message) {
  Data data = Json.decodeValue(message.body(), Data.class);
  vertx.eventBus().publish(
   "/token/" + data.getAddress(), message.body());
 }
}
```

Маршрутизатор


```
public class RouterVerticle extends AbstractVerticle {
 @Override
 public void start() {
  vertx.eventBus().consumer("router", this::router);
 }

 void router(Message<String> message) {
  Data data = Json.decodeValue(message.body(), Data.class);
  vertx.eventBus().publish(
   "/token/" + data.getAddress(), message.body());
 }
}
```

Маршрутизатор


```
public class RouterVerticle extends AbstractVerticle {
 @Override
 public void start() {
  vertx.eventBus().consumer("router", this::router);
 }

 void router(Message<String> message) {
  Data data = Json.decodeValue(message.body(), Data.class);
  vertx.eventBus().publish(
   "/token/" + data.getAddress(), message.body());
 }
}
```

Маршрутизатор


```
void createWebSocketServer(ServerWebSocket wsServer) {
 wsServer.frameHandler(wsFrame -> {


 vertx.eventBus().publish(
  "router", wsFrame.textData());
 });
}
```

Маршрутизатор


```
public class RouterVerticle extends AbstractVerticle {
 @Override
 public void start() {
  vertx.eventBus().consumer("router", this::router);
 }

 void router(Message<String> message) {
  Data data = Json.decodeValue(message.body(), Data.class);
  vertx.eventBus().publish(
   "/token/" + data.getAddress(), message.body());
 }
}
```

Маршрутизатор


```
public class RouterVerticle extends AbstractVerticle {
 @Override
 public void start() {
  vertx.eventBus().consumer("router", this::router);
 }

 void router(Message<String> message) {
  Data data = Json.decodeValue(message.body(), Data.class);
  vertx.eventBus().publish(
   "/token/" + data.getAddress(), message.body());
 }
}
```

Маршрутизатор

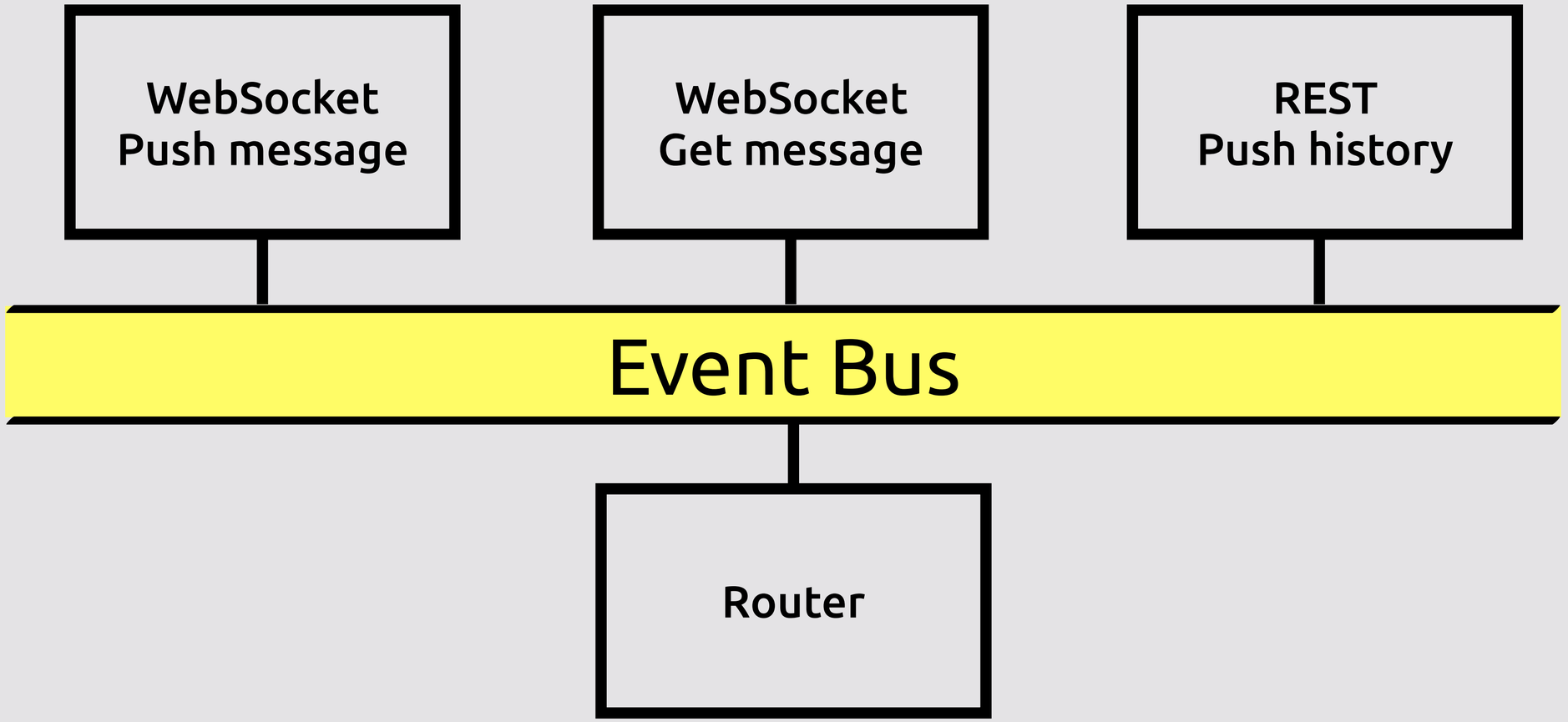

```
void createWebSocketServer(ServerWebSocket wsServer) {
 vertx.eventBus().<String>consumer(wsServer.path(), data -> {
  wsServer.writeFinalTextFrame(data.body());
  data.reply("ok");
 });
}
```

Продумывайте маршруты

Научились

- Первое приложение
- Блокирующие операции
- WebSocket
- Путь сообщения

Общая шина v2.1



Rest-клиент

```
curl \
-H "Content-Type: application/json" \
-X POST -d \
'{"text":"Message","address":"token"}' \
http://HOST/sendMessage
```

Отправка сообщений по REST


```
public class RestServerVerticle extends AbstractVerticle {
    @Override
    public void start() {
    }
}
```

Отправка сообщений по REST


```
public class RestServerVerticle extends AbstractVerticle {
    @Override
    public void start() {
    }
}
```

Отправка сообщений по REST


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
   .handler(request -> {
     vertx.eventBus().publish("router", request.getBodyAsString());
     request.response().end("ok");
   });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

Отправка сообщений по REST


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
   .handler(request -> {
     vertx.eventBus().publish("router", request.getBodyAsString());
     request.response().end("ok");
   });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

Отправка сообщений по REST


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
   .handler(request -> {
     vertx.eventBus().publish("router", request.getBodyAsString());
     request.response().end("ok");
   });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

Отправка сообщений по REST


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
    .handler(request -> {
      vertx.eventBus().publish("router", request.getBodyAsString());
      request.response().end("ok");
    });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

Отправка сообщений по REST


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
   .handler(request -> {
     vertx.eventBus().publish("router", request.getBodyAsString());
     request.response().end("ok");
   });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

Отправка сообщений по REST


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
   .handler(request -> {
     vertx.eventBus().publish("router", request.getBodyAsString());
     request.response().end("ok");
   });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

Отправка сообщений по REST


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
   .handler(request -> {
     vertx.eventBus().publish("router", request.getBodyAsString());
     request.response().end("ok");
   });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

Отправка сообщений по REST


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
   .handler(request -> {
     vertx.eventBus().publish("router", request.getBodyAsString());
     request.response().end("ok");
   });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

Отправка сообщений по REST


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
   .handler(request -> {
     vertx.eventBus().publish("router", request.getBodyAsString());
     request.response().end("ok");
   });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

Отправка сообщений по REST


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
   .handler(request -> {
     vertx.eventBus().publish("router", request.getBodyAsString());
     request.response().end("ok");
   });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

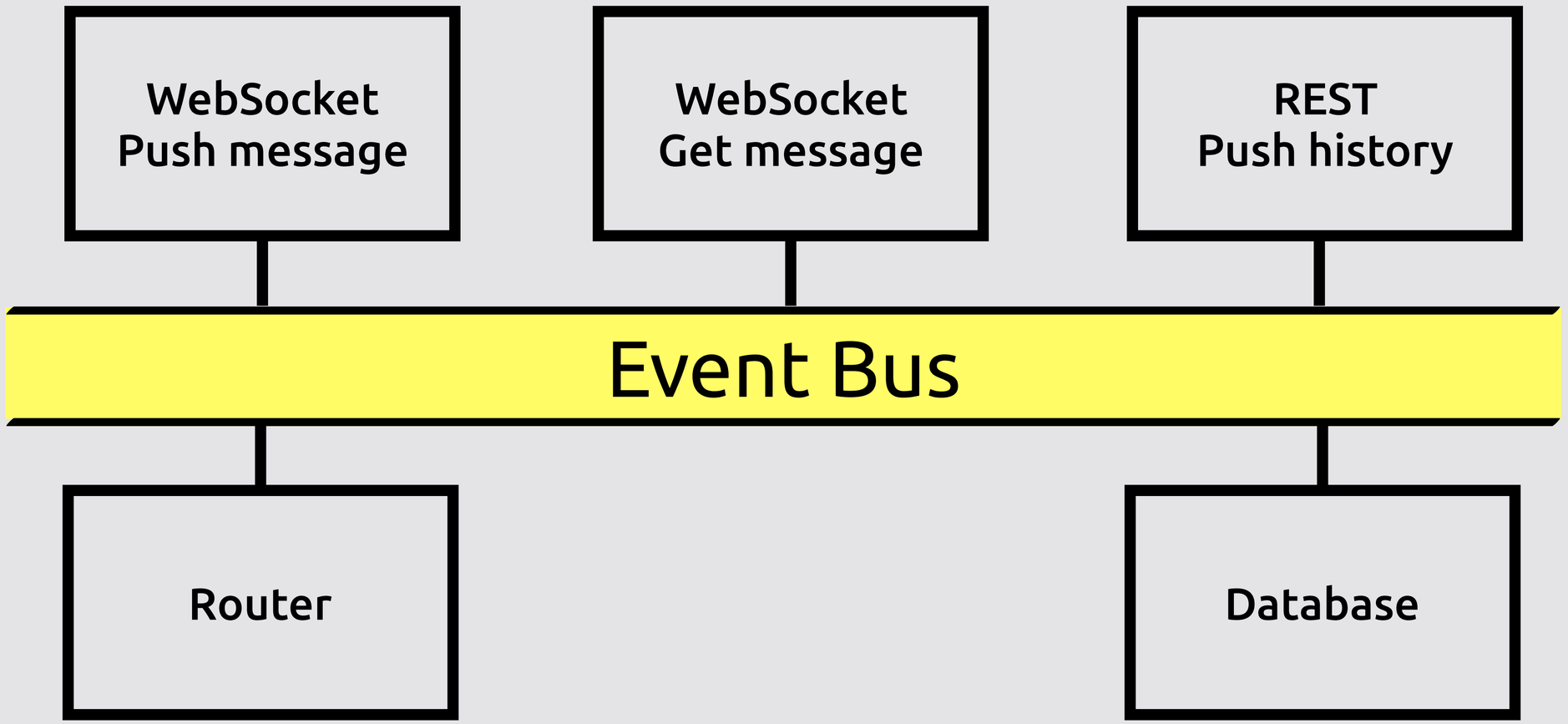
Отправка сообщений по REST

Demo

Научились

- Первое приложение
- Блокирующие операции
- WebSocket
- Путь сообщения
- REST

Общая шина v2.2



MongoDB


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
    client = MongoClient.createShared(vertx, new JsonObject()
      .put("db_name", "my_DB"));
    vertx.eventBus().consumer("router", this::saveDb);
  }
}
```

MongoDB


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
    client = MongoClient.createShared(vertx, new JsonObject()
      .put("db_name", "my_DB"));
    vertx.eventBus().consumer("router", this::saveDb);
  }
}
```

MongoDB


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
    client = MongoClient.createShared(vertx, new JsonObject()
      .put("db_name", "my_DB"));
    vertx.eventBus().consumer("router", this::saveDb);
  }
}
```

MongoDB


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
    client = MongoClient.createShared(vertx, new JsonObject()
      .put("db_name", "my_DB"));
    vertx.eventBus().consumer("router", this::saveDb);
  }
}
```

MongoDB


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
    client = MongoClient.createShared(vertx, new JsonObject()
      .put("db_name", "my_DB"));
    vertx.eventBus().consumer("router", this::saveDb);
  }
}
```

MongoDB


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
    client = MongoClient.createShared(vertx, new JsonObject()
      .put("db_name", "my_DB"));
    vertx.eventBus().consumer("router", this::saveDb);
  }
}
```

MongoDB


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
    client = MongoClient.createShared(vertx, new JsonObject()
      .put("db_name", "my_DB"));
    vertx.eventBus().consumer("router", this::saveDb);
  }
}
```

MongoDB


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
    client = MongoClient.createShared(vertx, new JsonObject()
      .put("db_name", "my_DB"));
    vertx.eventBus().consumer("router", this::saveDb);
  }
}

void saveDb(Message<String> message) {
}
```

MongoDB


```
void saveDb(Message<String> message) {
  client.insert("message",
    new JsonObject(message.body()), this::handlerSaved);
}

void handlerSaved(AsyncResult<String> stringAsyncResult) {
  if (stringAsyncResult.succeeded()) {
    System.out.println("MongoDB save: "
      + stringAsyncResult.result());
  } else {
    System.out.println("ERROR MongoDB: "
      + stringAsyncResult.cause());
  }
}
```

MongoDB


```
void saveDb(Message<String> message) {
  client.insert("message",
    new JsonObject(message.body()), this::handlerSaved);
}

void handlerSaved(AsyncResult<String> stringAsyncResult) {
  if (stringAsyncResult.succeeded()) {
    System.out.println("MongoDB save: "
      + stringAsyncResult.result());
  } else {
    System.out.println("ERROR MongoDB: "
      + stringAsyncResult.cause());
  }
}
```

MongoDB


```
void saveDb(Message<String> message) {
  client.insert("message",
    new JsonObject(message.body()), this::handlerSaved);
}

void handlerSaved(AsyncResult<String> stringAsyncResult) {
  if (stringAsyncResult.succeeded()) {
    System.out.println("MongoDB save: "
      + stringAsyncResult.result());
  } else {
    System.out.println("ERROR MongoDB: "
      + stringAsyncResult.cause());
  }
}
```

MongoDB


```
void createWebSocketServer(ServerWebSocket wsServer) {
 wsServer.frameHandler(wsFrame -> {


 vertx.eventBus().publish(
  "router", wsFrame.textData());
 });
}
```

MongoDB


```
void saveDb(Message<String> message) {
  client.insert("message",
    new JsonObject(message.body()), this::handlerSaved);
}

void handlerSaved(AsyncResult<String> stringAsyncResult) {
  if (stringAsyncResult.succeeded()) {
    System.out.println("MongoDB save: "
      + stringAsyncResult.result());
  } else {
    System.out.println("ERROR MongoDB: "
      + stringAsyncResult.cause());
  }
}
```

MongoDB


```
void saveDb(Message<String> message) {
  client.insert("message",
    new JsonObject(message.body()), this::handlerSaved);
}

void handlerSaved(AsyncResult<String> stringAsyncResult) {
  if (stringAsyncResult.succeeded()) {
    System.out.println("MongoDB save: "
      + stringAsyncResult.result());
  } else {
    System.out.println("ERROR MongoDB: "
      + stringAsyncResult.cause());
  }
}
```

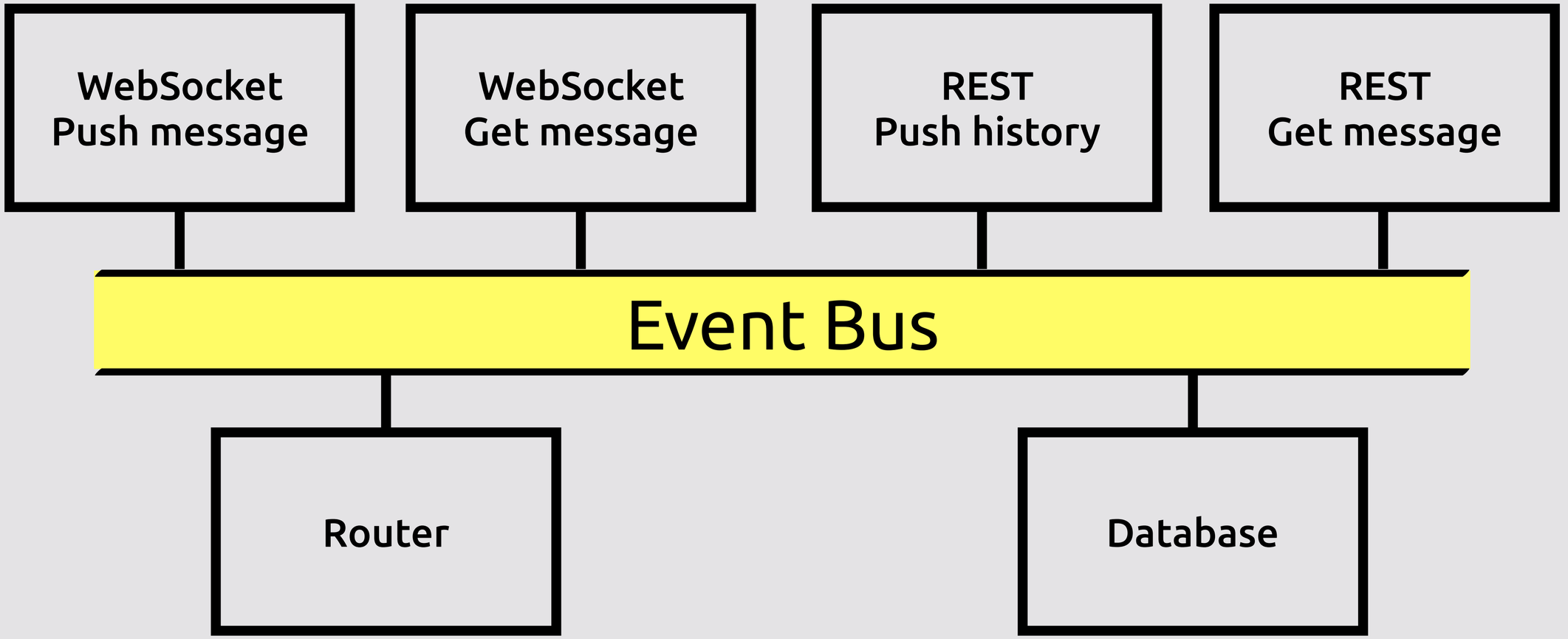
MongoDB


```
void saveDb(Message<String> message) {
  client.insert("message",
    new JsonObject(message.body()), this::handlerSaved);
}

void handlerSaved(AsyncResult<String> stringAsyncResult) {
  if (stringAsyncResult.succeeded()) {
    System.out.println("MongoDB save: "
      + stringAsyncResult.result());
  } else {
    System.out.println("ERROR MongoDB: "
      + stringAsyncResult.cause());
  }
}
```

Научились

- Первое приложение
- Блокирующие операции
- WebSocket
- Путь сообщения
- REST
- База данных

Общая шина v2.3



Rest-клиент

```
curl http://HOST/getHistory
```

Прием сообщений по REST


```
public class RestServerVerticle extends AbstractVerticle {
    @Override
    public void start() {
    }
}
```

Прием сообщений по REST


```
@Override
public void start() {
// . . .
  httpRouter.get("/getHistory")
  .handler(request ->
    vertx.eventBus().send(
      "getHistory", request.getBodyAsString(),
        result ->
          request.response().end(
            result.result().body().toString())));
// . . .
}
```

Прием сообщений по REST


```
@Override
public void start() {
// . . .
  httpRouter.get("/getHistory")
  .handler(request ->
    vertx.eventBus().send(
      "getHistory", request.getBodyAsString(),
        result ->
          request.response().end(
            result.result().body().toString())));
// . . .
}
```

Прием сообщений по REST


```
@Override
public void start() {
// . . .
  httpRouter.get("/getHistory")
  .handler(request ->
    vertx.eventBus().send(
      "getHistory", request.getBodyAsString(),
        result ->
          request.response().end(
            result.result().body().toString())));
// . . .
}
```

Прием сообщений по REST


```
@Override
public void start() {
// . . .
  httpRouter.get("/getHistory")
  .handler(request ->
    vertx.eventBus().send(
      "getHistory", request.getBodyAsString(),
        result ->
          request.response().end(
            result.result().body().toString())));
// . . .
}
```

Прием сообщений по REST


```
@Override
public void start() {
// . . .
  httpRouter.get("/getHistory")
  .handler(request ->
    vertx.eventBus().send(
      "getHistory", request.getBodyAsString(),
        result ->
          request.response().end(
            result.result().body().toString())));
// . . .
}
```

Прием сообщений по REST


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
  // . . .
  vertx.eventBus().consumer("getHistory", this::getHistory);
  }
}
void getHistory(Message<String> message) {
  client.find("message", new JsonObject(),
    result -> message.reply(
      Json.encode(result.result())))
    );
}
```

Прием сообщений по REST


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
  // . . .
  vertx.eventBus().consumer("getHistory", this::getHistory);
  }
}
void getHistory(Message<String> message) {
  client.find("message", new JsonObject(),
    result -> message.reply(
      Json.encode(result.result()))
    );
}
```

Прием сообщений по REST


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
  // . . .
  vertx.eventBus().consumer("getHistory", this::getHistory);
  }
}
void getHistory(Message<String> message) {
  client.find("message", new JsonObject(),
    result -> message.reply(
      Json.encode(result.result()))
    );
}
```

Прием сообщений по REST


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
  // . . .
  vertx.eventBus().consumer("getHistory", this::getHistory);
  }
}
void getHistory(Message<String> message) {
  client.find("message", new JsonObject(),
    result -> message.reply(
      Json.encode(result.result()))
    );
}
```

Прием сообщений по REST


```
@Override
public void start() {
// . . .
  httpRouter.get("/getHistory")
  .handler(request ->
    vertx.eventBus().send(
      "getHistory", request.getBodyAsString(),
        result ->
          request.response().end(
            result.result().body().toString())));
// . . .
}
```

Прием сообщений по REST

Demo

Kubernetes описание

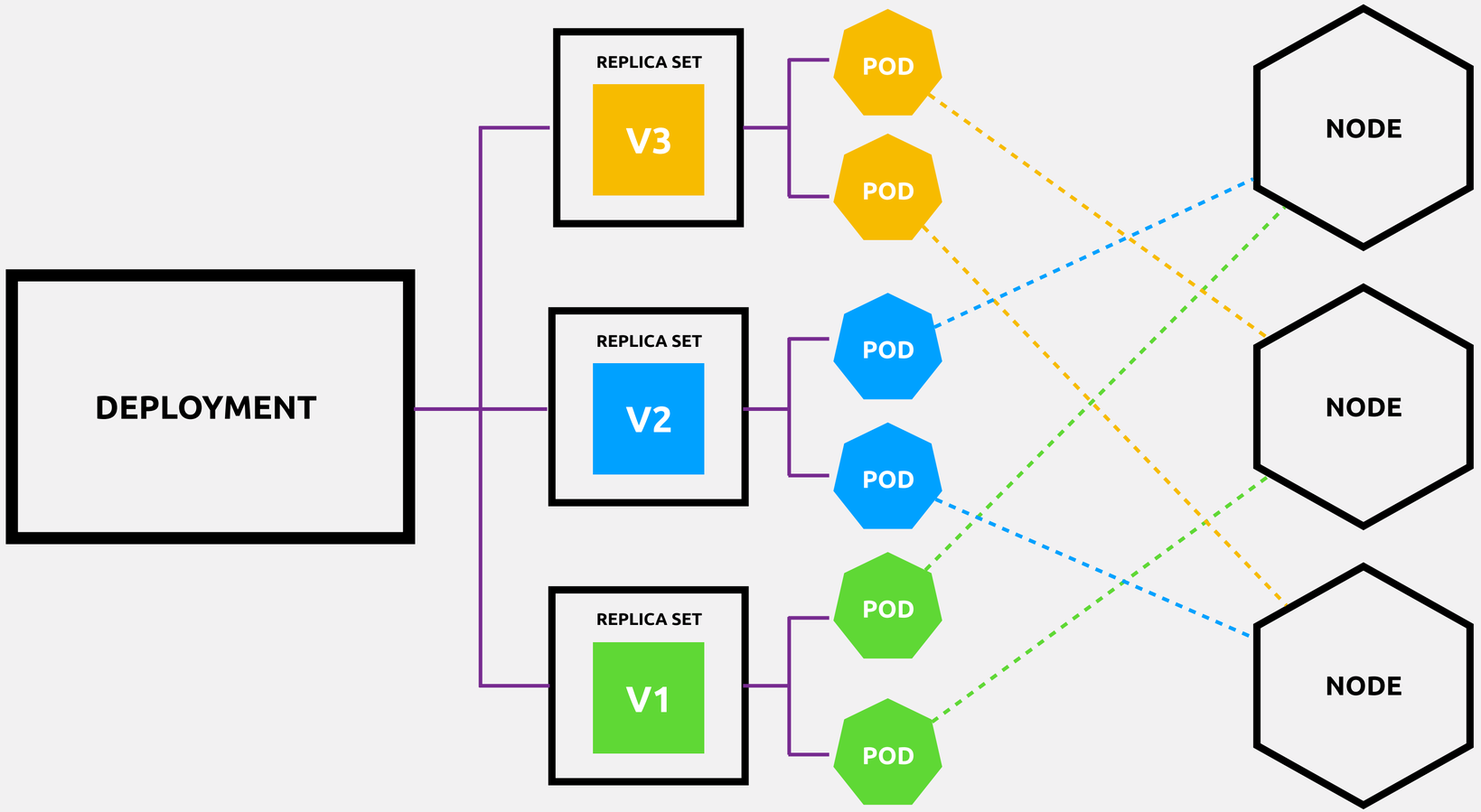
- С греческого — «Рулевой»
- Первоначальный разработчик — Google
- Управляет Docker контейнерами

Kubernetes

Задача — автоматизировать работу с Docker контейнерами:

- Установка
- Обновление
- Масштабирование
- Возврат к старой версии
- Мониторинг
- Логирование

Что такое Deployment



Deploy


```
public class Starter {

  public static void main(String[] args) {
    Vertx.vertx().deployVerticle(new WsServerVerticle());
    Vertx.vertx().deployVerticle(new RestServerVerticle());
    Vertx.vertx().deployVerticle(new RouterVerticle());
    Vertx.vertx().deployVerticle(new MongoDbVerticle());
  }
}
```

Deploy


```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jbreak-chat
spec:
  selector:
    matchLabels:
      run: jbreak-chat
  replicas: 1
  template:
    metadata:
      labels:
        run: jbreak-chat
    spec:
      containers:
      - name: jbreak-chat
        image: dzx912/jbreak-chat:1
        ports:
        - containerPort: 8080
        - containerPort: 8081
```

Deploy


```
---
apiVersion: v1
kind: Service
metadata:
  name: jbreak-chat
  labels:
    run: jbreak-chat
spec:
  selector:
    run: jbreak-chat-server
  ports:
  - name: websocket
    port: 8080
    nodePort: 30080
  - name: http
    port: 8081
    nodePort: 30081
  type: LoadBalancer
```

Deploy


```
kubectl apply -f chat.yaml
```

Deploy

Demo

Cluster Manager в Vert.x

hazelcast

Infinispan

Apache ZooKeeper™

apache Ignite

Cluster в Vert.x


```
IgniteConfiguration getIgniteConfiguration() {
    TcpDiscoverySpi spi = new TcpDiscoverySpi();
    TcpDiscoveryKubernetesIpFinder ipFinder =
            new TcpDiscoveryKubernetesIpFinder();

    ipFinder.setServiceName("jbreak-chat");

    spi.setIpFinder(ipFinder);
    IgniteConfiguration cfg = new IgniteConfiguration();
    cfg.setDiscoverySpi(spi);
    return cfg;
}
```

Cluster в Vert.x


```
---
apiVersion: v1
kind: Service
metadata:
  name: jbreak-chat
. . . .
```

Cluster в Vert.x


```
IgniteConfiguration getIgniteConfiguration() {
    TcpDiscoverySpi spi = new TcpDiscoverySpi();
    TcpDiscoveryKubernetesIpFinder ipFinder =
            new TcpDiscoveryKubernetesIpFinder();

    ipFinder.setServiceName("jbreak-chat");

    spi.setIpFinder(ipFinder);
    IgniteConfiguration cfg = new IgniteConfiguration();
    cfg.setDiscoverySpi(spi);
    return cfg;
}
```

Cluster в Vert.x


```
void setupCluster() {
    ClusterManager clusterManager =
        new IgniteClusterManager(getIgniteConfiguration());

    VertxOptions options = new VertxOptions()
        .setClustered(true)
        .setClusterManager(clusterManager)
        .setClusterHost(getMyIp());

    Vertx.clusteredVertx(options, res -> {
        if (res.succeeded()) {
            Vertx vertx = res.result();
            deploy(vertx);
        }
    });
}
```

Cluster в Vert.x


```
void setupCluster() {
    ClusterManager clusterManager =
        new IgniteClusterManager(getIgniteConfiguration());

    VertxOptions options = new VertxOptions()
        .setClustered(true)
        .setClusterManager(clusterManager)
        .setClusterHost(getMyIp());

    Vertx.clusteredVertx(options, res -> {
        if (res.succeeded()) {
            Vertx vertx = res.result();
            deploy(vertx);
        }
    });
}
```

Cluster в Vert.x


```
void setupCluster() {
    ClusterManager clusterManager =
        new IgniteClusterManager(getIgniteConfiguration());

    VertxOptions options = new VertxOptions()
        .setClustered(true)
        .setClusterManager(clusterManager)
        .setClusterHost(getMyIp());

    Vertx.clusteredVertx(options, res -> {
        if (res.succeeded()) {
            Vertx vertx = res.result();
            deploy(vertx);
        }
    });
}
```

Cluster в Vert.x


```
void setupCluster() {
    ClusterManager clusterManager =
        new IgniteClusterManager(getIgniteConfiguration());

    VertxOptions options = new VertxOptions()
        .setClustered(true)
        .setClusterManager(clusterManager)
        .setClusterHost(getMyIp());

    Vertx.clusteredVertx(options, res -> {
        if (res.succeeded()) {
            Vertx vertx = res.result();
            deploy(vertx);
        }
    });
}
```

Cluster в Vert.x


```
void setupCluster() {
    ClusterManager clusterManager =
        new IgniteClusterManager(getIgniteConfiguration());

    VertxOptions options = new VertxOptions()
        .setClustered(true)
        .setClusterManager(clusterManager)
        .setClusterHost(getMyIp());

    Vertx.clusteredVertx(options, res -> {
        if (res.succeeded()) {
            Vertx vertx = res.result();
            deploy(vertx);
        }
    });
}
```

Cluster в Vert.x


```
void setupCluster() {
    ClusterManager clusterManager =
        new IgniteClusterManager(getIgniteConfiguration());

    VertxOptions options = new VertxOptions()
        .setClustered(true)
        .setClusterManager(clusterManager)
        .setClusterHost(getMyIp());

    Vertx.clusteredVertx(options, res -> {
        if (res.succeeded()) {
            Vertx vertx = res.result();
            deploy(vertx);
        }
    });
}
```

Cluster в Vert.x


```
void deploy(Vertx vertx) {
    vertx.deployVerticle(new WsServerVerticle());
    vertx.deployVerticle(new RestServerVerticle());
    vertx.deployVerticle(new RouterVerticle());
    vertx.deployVerticle(new MongoDbVerticle());
}
```

Научились

- Первое приложение
- Блокирующие операции
- WebSocket
- Путь сообщения
- REST
- База данных
- Кластеризация

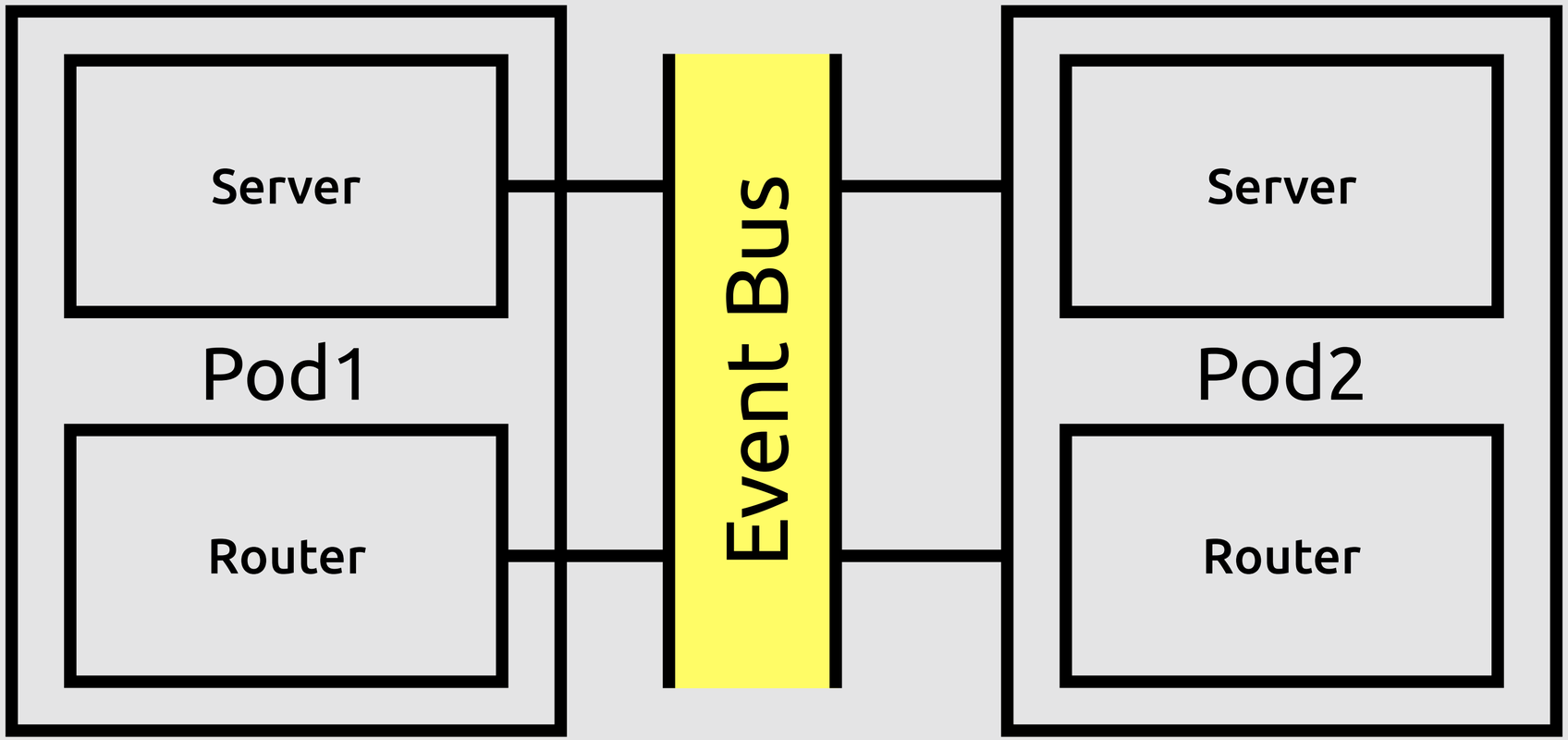
Demo

Продумывайте маршруты

Научились

- Первое приложение
- Блокирующие операции
- WebSocket
- ~~Путь сообщения~~
- REST
- База данных
- Кластеризация

Cluster
в Vert.x



Cluster в Vert.x


```
void deploy(Vertx vertx) {
    vertx.deployVerticle(new WsServerVerticle());
    vertx.deployVerticle(new RestServerVerticle());
}
```

Cluster в Vert.x


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jbreak-chat-server
. . . .
```

Cluster в Vert.x


```
void deploy(Vertx vertx) {
    vertx.deployVerticle(new RouterVerticle());
}
```

Cluster в Vert.x


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jbreak-chat-router
. . . .
```

Cluster в Vert.x


```
void deploy(Vertx vertx) {
    vertx.deployVerticle(new MongoDbVerticle());
}
```

Cluster в Vert.x


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jbreak-chat-mongo
. . . .
```

Cluster в Vert.x


```
void createWebSocketServer(ServerWebSocket wsServer) {
 wsServer.frameHandler(wsFrame -> {


   vertx.eventBus().publish(
   "router", wsFrame.textData());
 });
}
```

Cluster в Vert.x


```
void createWebSocketServer(ServerWebSocket wsServer) {
 wsServer.frameHandler(wsFrame -> {


   vertx.eventBus().send(
   "router", wsFrame.textData());
 });
}
```

Cluster в Vert.x


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
   .handler(request -> {
     vertx.eventBus().publish("router", request.getBodyAsString());
     request.response().end("ok");
   });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

Cluster в Vert.x


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
   .handler(request -> {
     vertx.eventBus().send("router", request.getBodyAsString());
     request.response().end("ok");
   });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

Cluster в Vert.x


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
    client = MongoClient.createShared(vertx, new JsonObject()
      .put("db_name", "my_DB"));
    vertx.eventBus().consumer("router", this::saveDb);
  }
}
```

Cluster в Vert.x


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
    client = MongoClient.createShared(vertx, new JsonObject()
      .put("db_name", "my_DB"));
    vertx.eventBus().consumer("database.save", this::saveDb);
  }
}
```

Cluster в Vert.x


```
public class RouterVerticle extends AbstractVerticle {
 void router(Message<String> message) {
  Data data = Json.decodeValue(message.body(), Data.class);
  vertx.eventBus().publish(
   "/token/" + data.getAddress(), message.body());

  vertx.eventBus().send("database.save", message.body());
 }
}
```

Demo

Научились

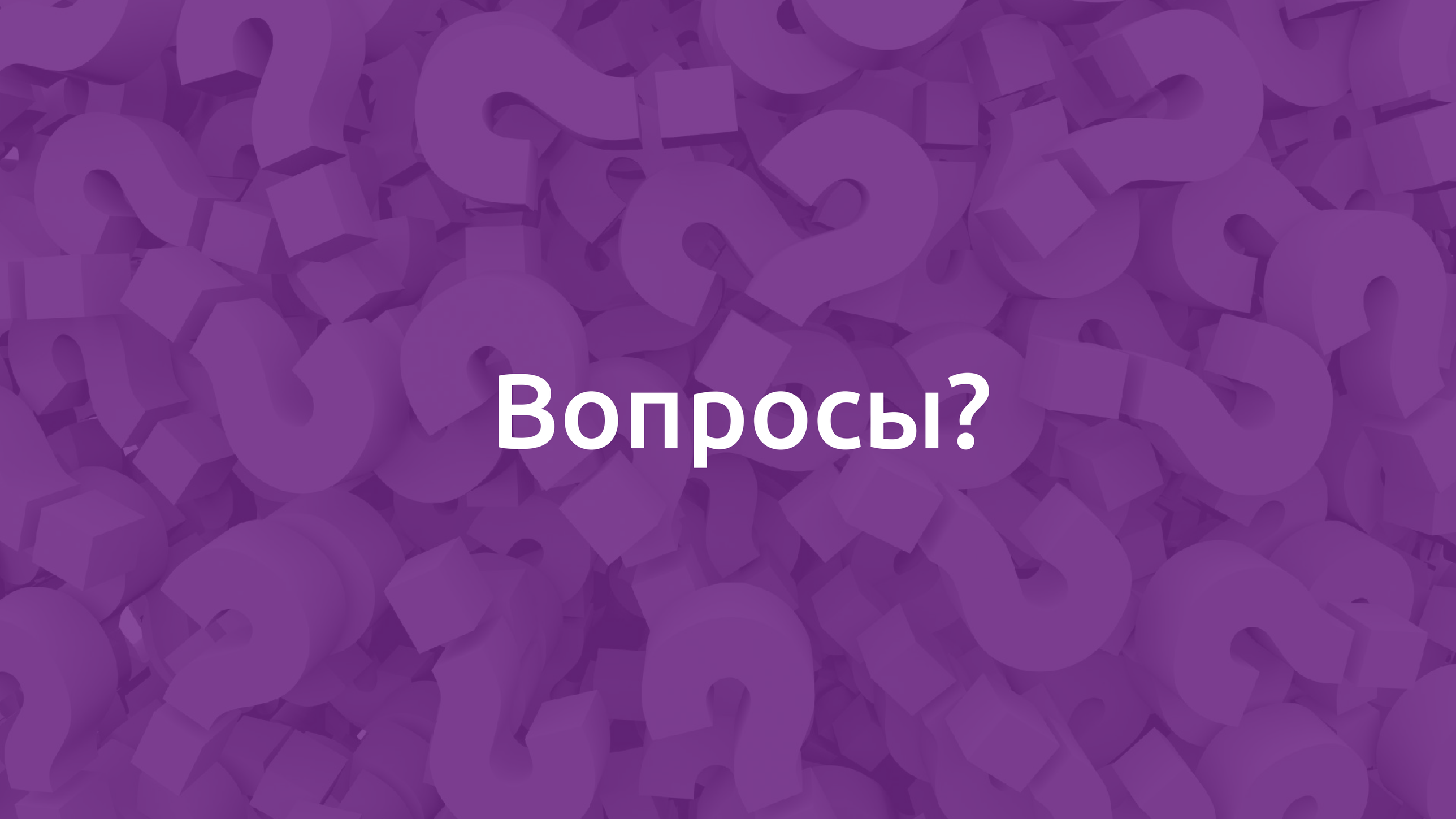
- Первое приложение
- Блокирующие операции
- WebSocket
- ~~Путь сообщения~~
- REST
- База данных
- Кластеризация
- Пусть сообщения в кластере

Чистим за собой

```
kubectl delete -f chat.yaml
```

Выученные уроки

- Не блокировать Event Bus
- 1 функция — 1 модуль
- Продумываем маршруты


Вопросы?

Сбербанк Онлайн

Ленок Антон

Серверная разработка

Проект: Диалоги
в Сбербанк Онлайн

AILenok.SBT@sberbank.ru