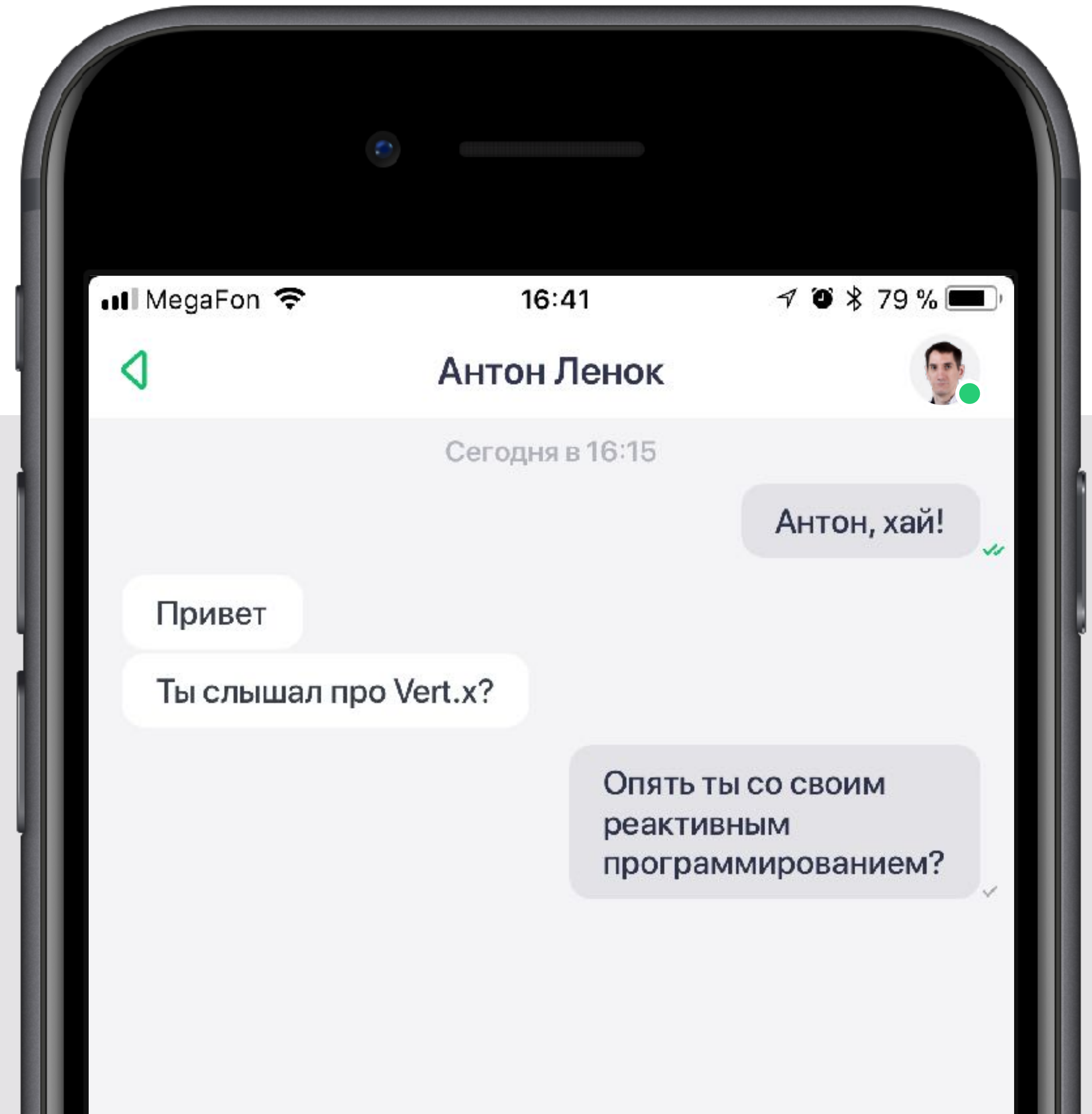


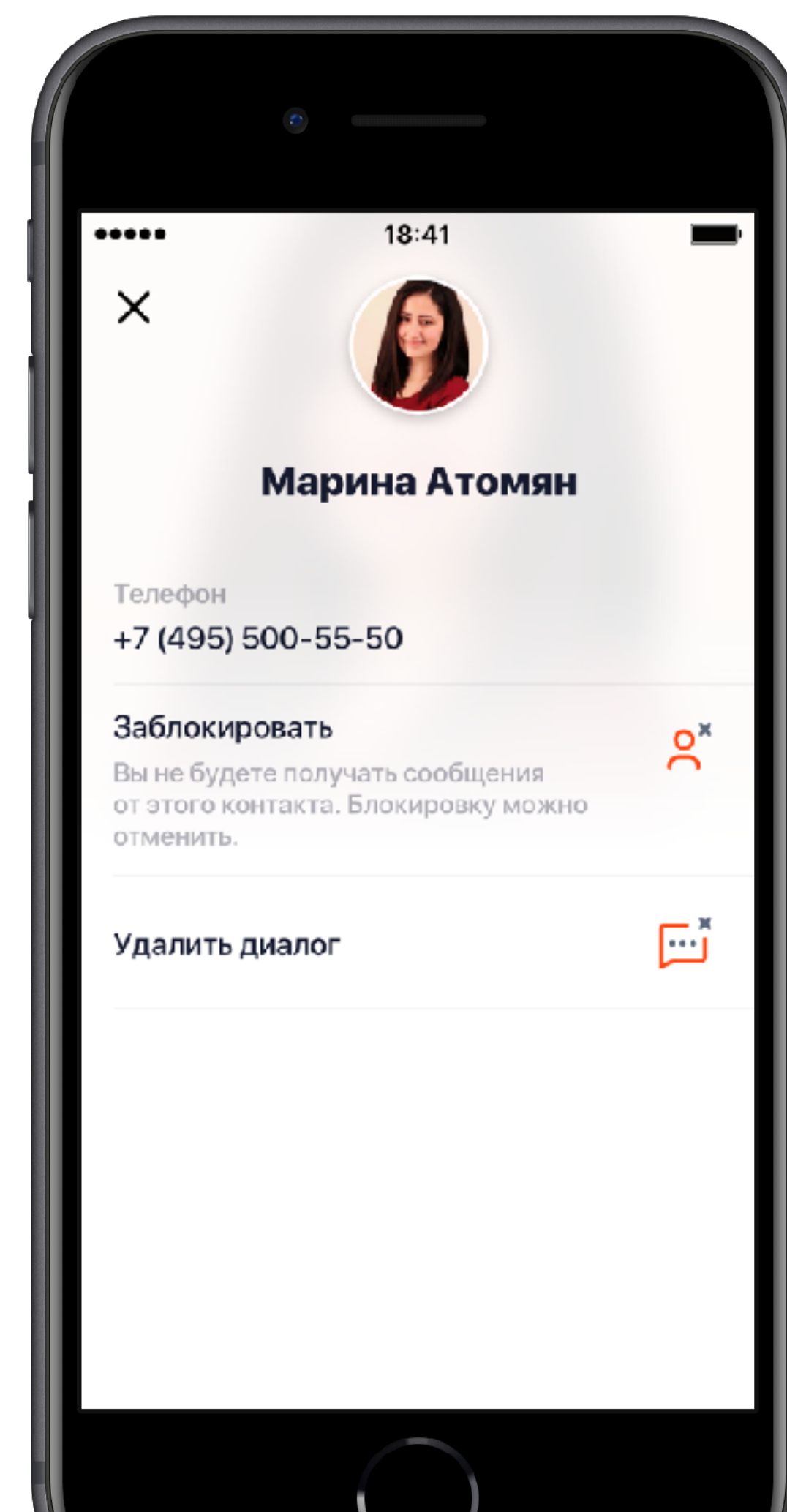
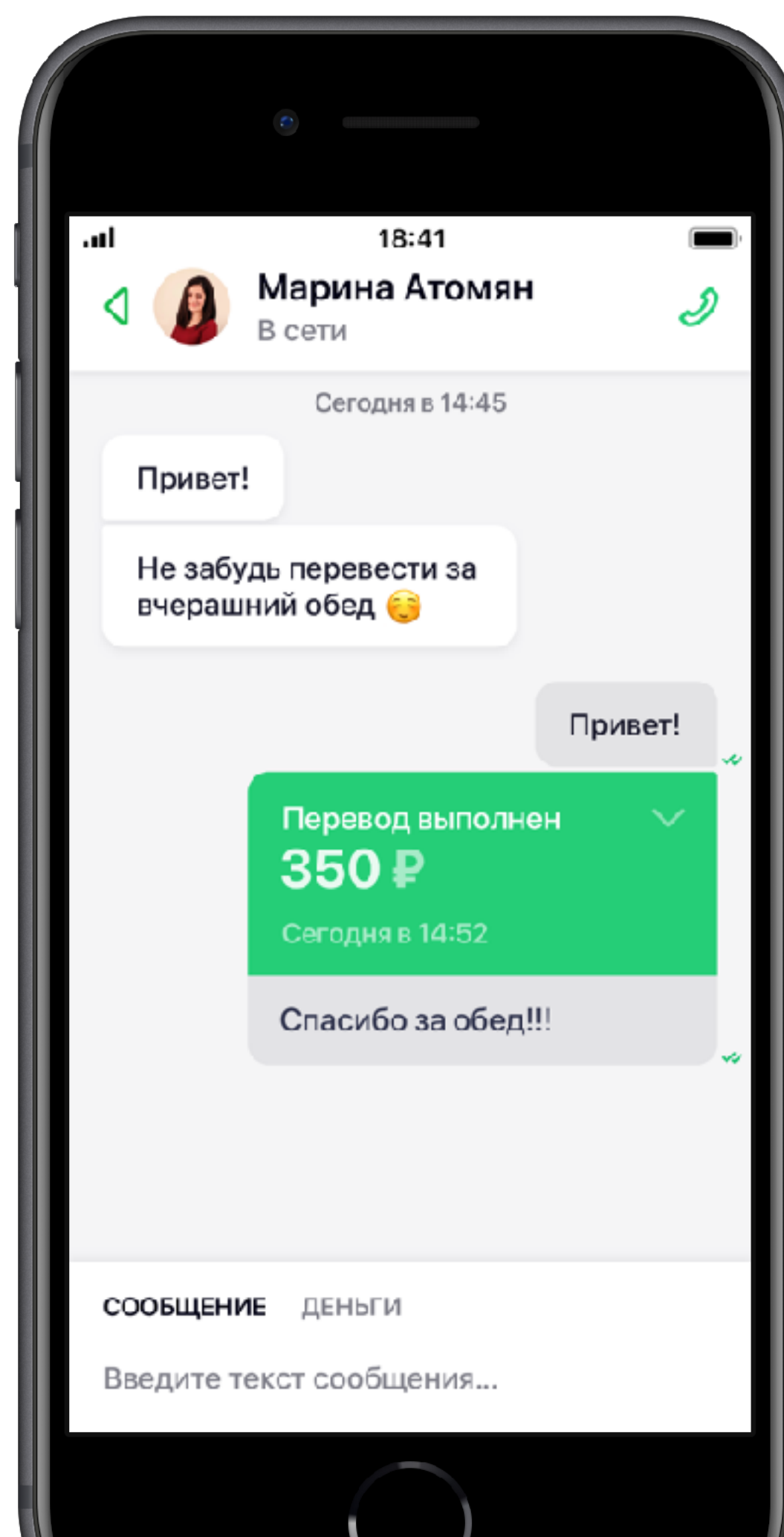
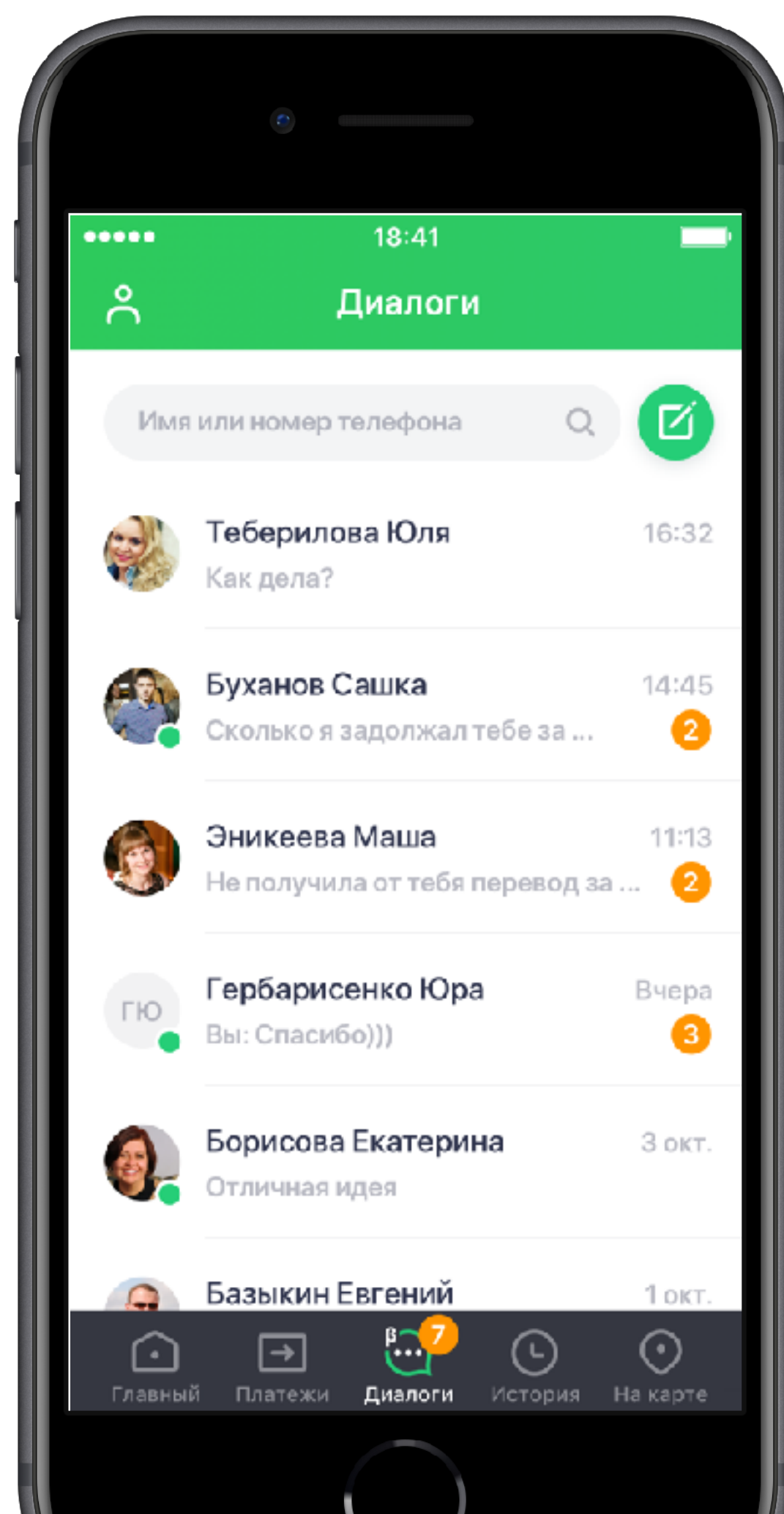


Сбербанк  
Онлайн

# 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

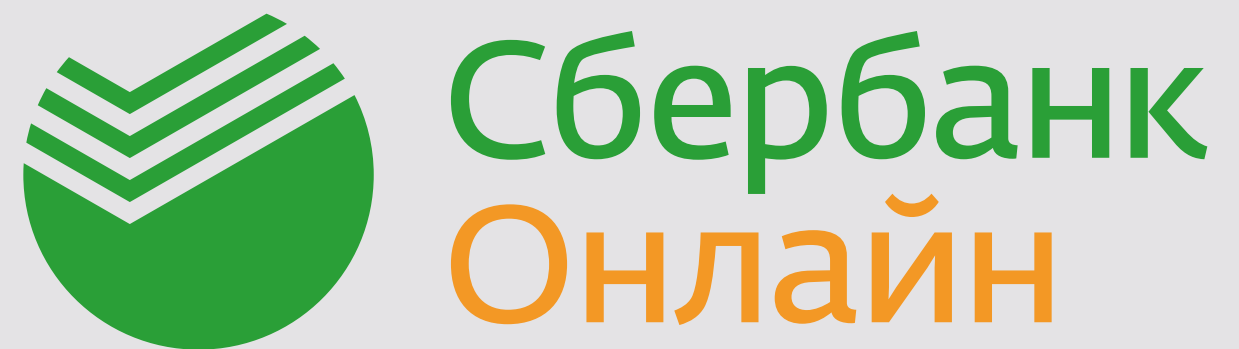
Че.

СООБЩЕНИЕ ДЕНЬГИ

Введите текст сообщения...







# Ленок Антон

Серверная разработка

Проект: Диалоги  
в Сбербанк Онлайн

[AlLenok.SBT@sberbank.ru](mailto:AlLenok.SBT@sberbank.ru)





# | Задача



Прием и отправка сообщений



Скорость отправки > приёма сообщения <1 секунды



В начале 10 человек онлайн

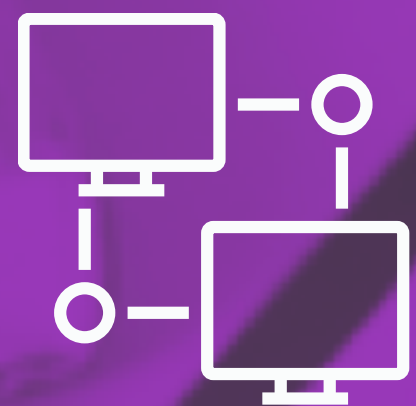


Целевое 100 000 человек онлайн



Быстрое добавление функций

# | Решение



WebSocket



Общая шина на сервере

# Vert.x описание

The logo for Vert.x, featuring the word "VERT." in a dark teal color and ".X" in a purple color, all in a bold, sans-serif font.

- 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

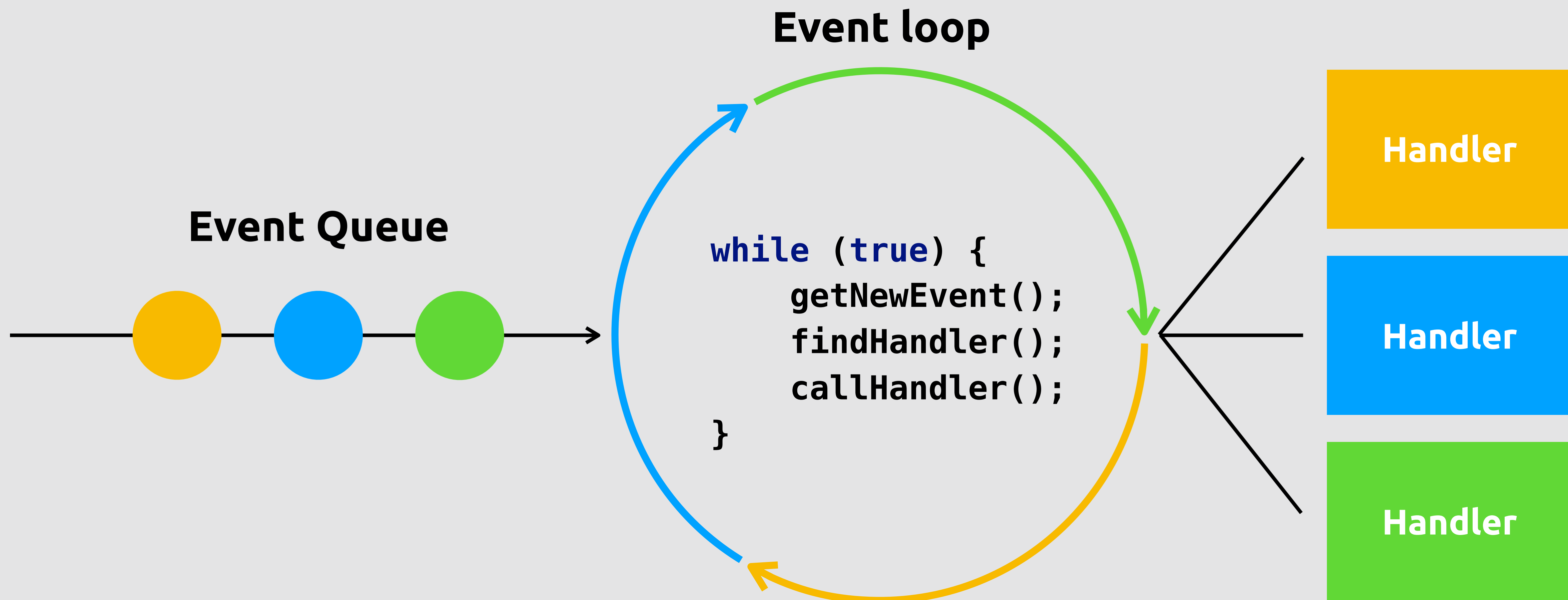
## Подробно:

- Принципы реактивного программирования
- Vert.x

## Не концентрируемся:

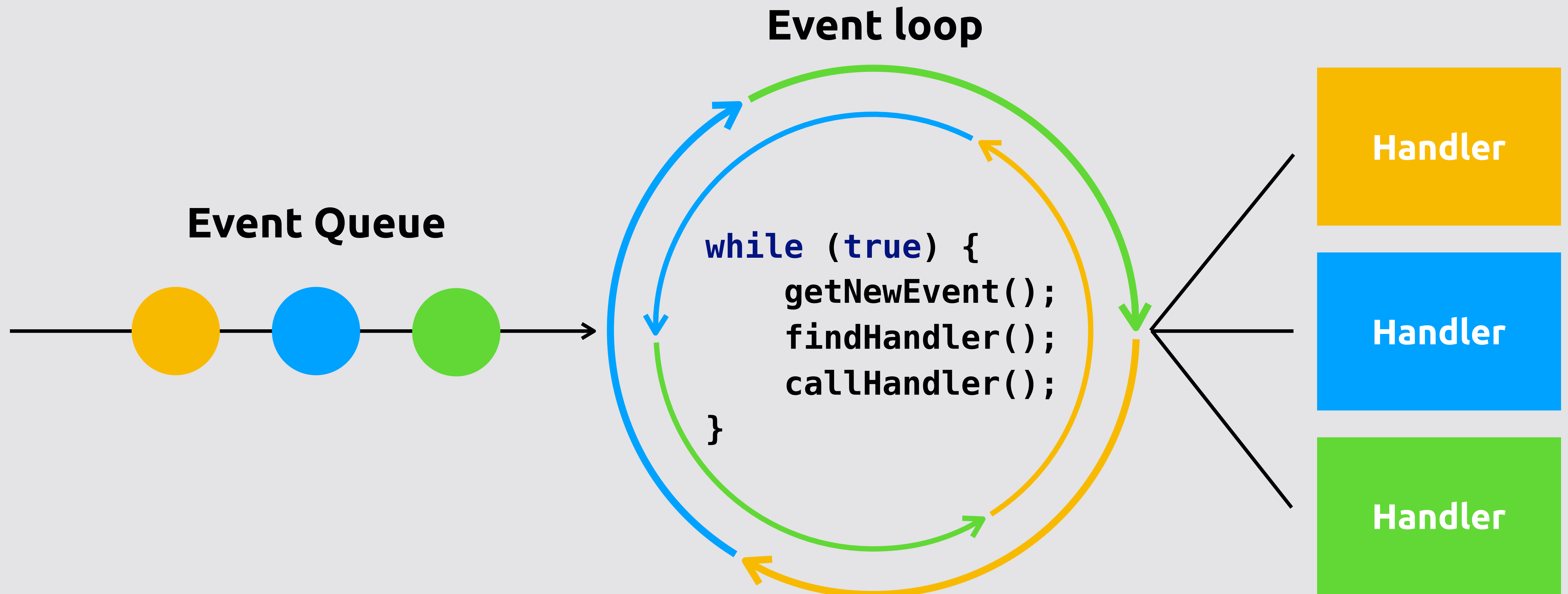
- Kubernetes
- Mongo DB

# Реактивное программирование





# | Multi-Reactor



# Первое приложение

1 / 9

```
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

2 / 9

```
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

3 / 9

```
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

4 / 9

```
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

5 / 9

```
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

6 / 9

```
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

7 / 9

```
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

8 / 9

```
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

9 / 9

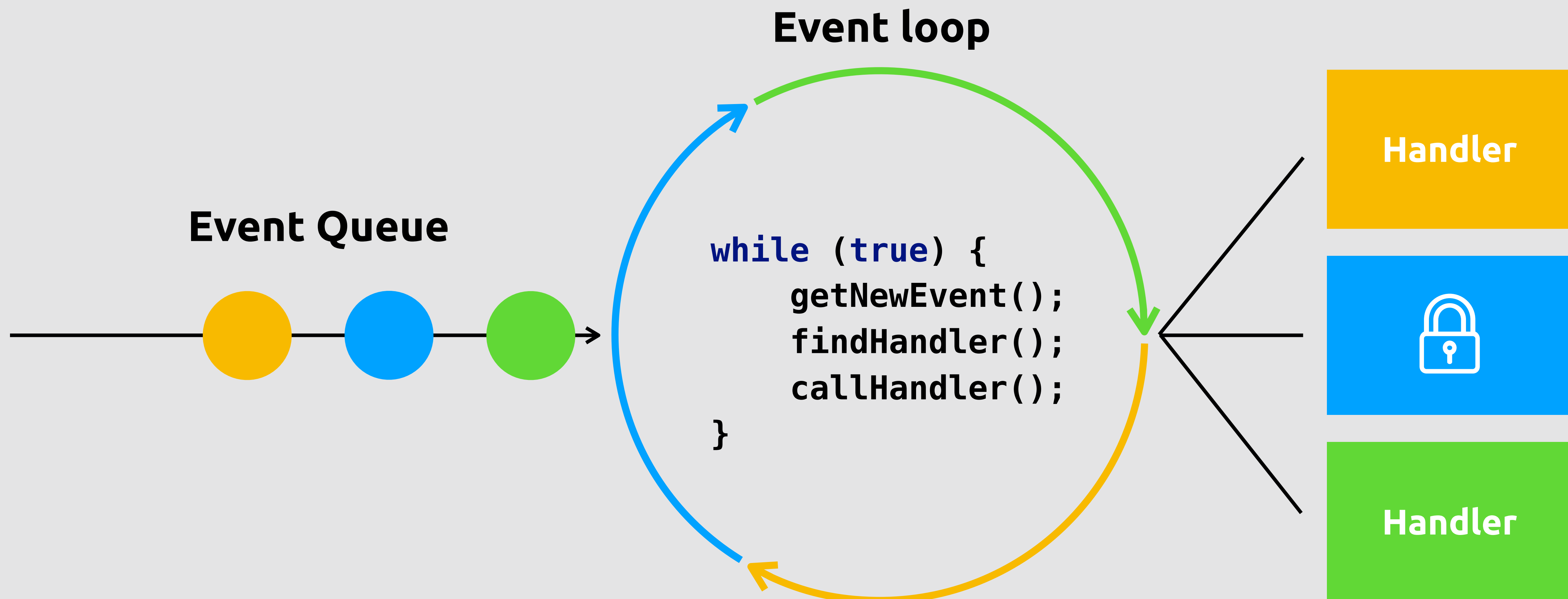
```
import io.vertx.core.Vertx;
import org.example.verticle>HelloVerticle;

public class Starter {

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

## ■ Первое приложение

# Блокирующие операции



# Блокирующие операции

1 / 3

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

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





# Блокирующие операции

2 / 3

```
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

3 / 3

```
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()));
}
```



- Первое приложение
- Блокирующие операции

# I Протокол

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



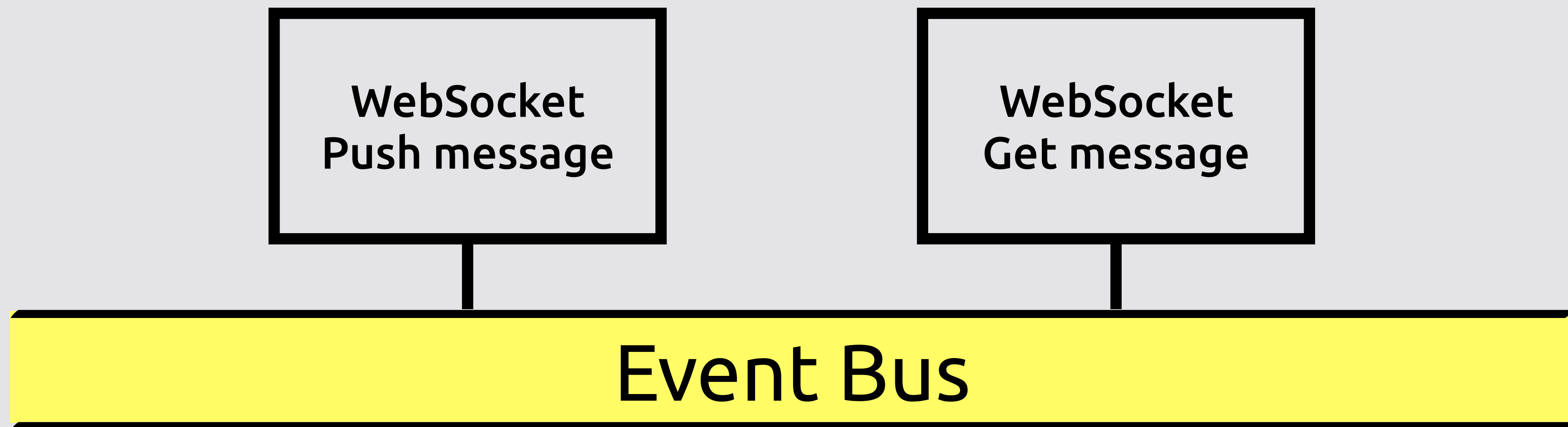
## | Клиент

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

## | Клиент

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

# | Общая шина v1.0



# Отправка сообщений

1 / 20

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



# Отправка сообщений

2 / 20

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

# | Отправка сообщений

3 / 20

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

# | Отправка сообщений

4 / 20

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

# Отправка сообщений

5 / 20

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

void createWebSocketServer
    (ServerWebSocket wsServer) {
}
```

# Отправка сообщений

6 / 20

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



# Отправка сообщений

7 / 20

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

# Отправка сообщений

8 / 20

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

# Отправка сообщений

9 / 20

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

# Отправка сообщений

10 / 20

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

# Отправка сообщений

11 / 20

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



# | Прием сообщений

12 / 20

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

# Прием сообщений

13 / 20

```
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

14 / 20

```
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

15 / 20

```
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

16 / 20

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

# Прием сообщений

17 / 20

```
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

18 / 20

```
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

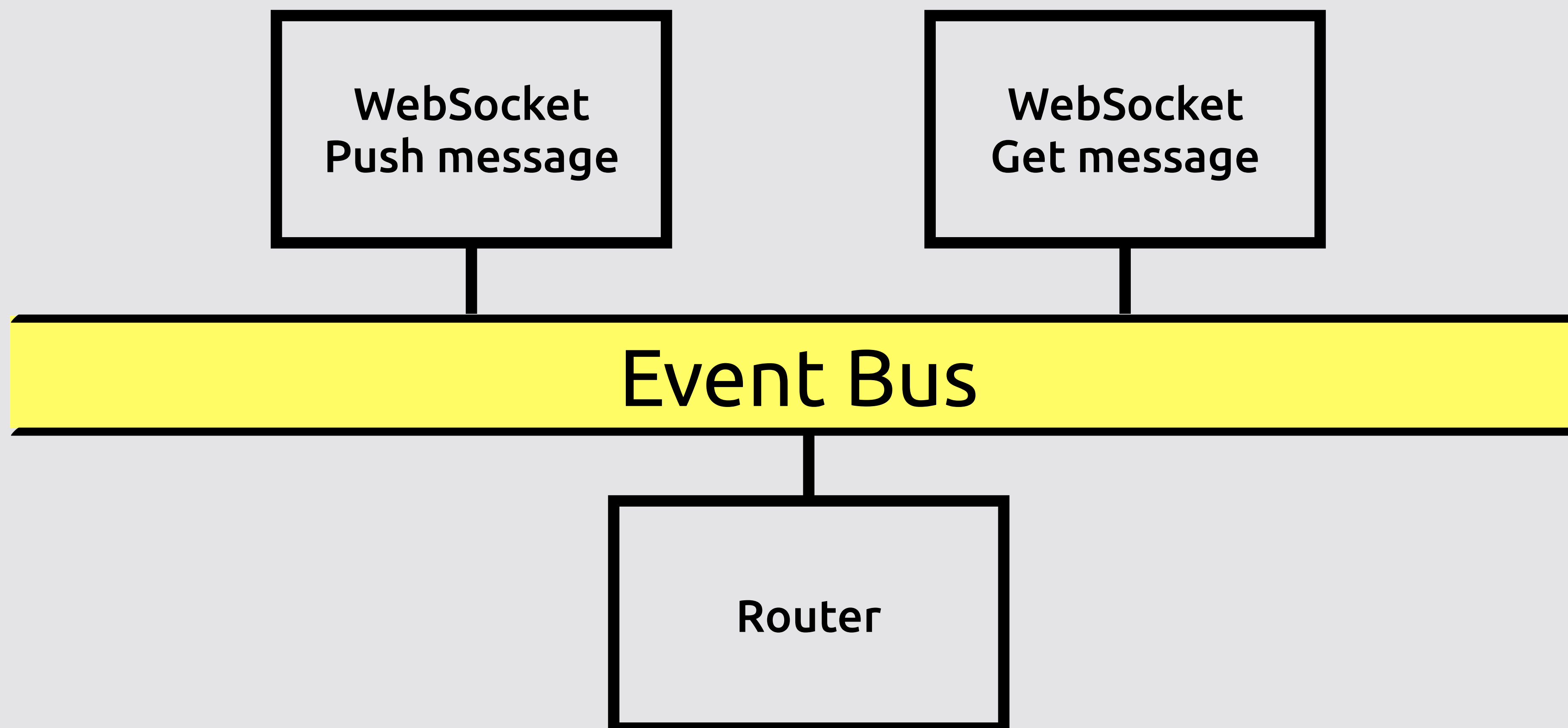
19 / 20

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

Demo

- Первое приложение
- Блокирующие операции
- WebSocket

# | Общая шина v2.0



# | Маршрутизатор

1 / 9

```
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

2 / 9

```
void createWebSocketServer(ServerWebSocket wsServer) {  
    wsServer.frameHandler(wsFrame -> {  
  
        vertx.eventBus().publish(  
            "router", wsFrame.textData());  
    });  
}
```



# | Маршрутизатор

3 / 9

```
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

4 / 9

```
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());
    }
}
```

# I Маршрутизатор

5 / 9

```
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

6 / 9

```
void createWebSocketServer(ServerWebSocket wsServer) {  
    wsServer.frameHandler(wsFrame -> {  
  
        vertx.eventBus().publish(  
            "router", wsFrame.textData());  
        });  
    }  
}
```

# | Маршрутизатор

7 / 9

```
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

8 / 9

```
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

9 / 9

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



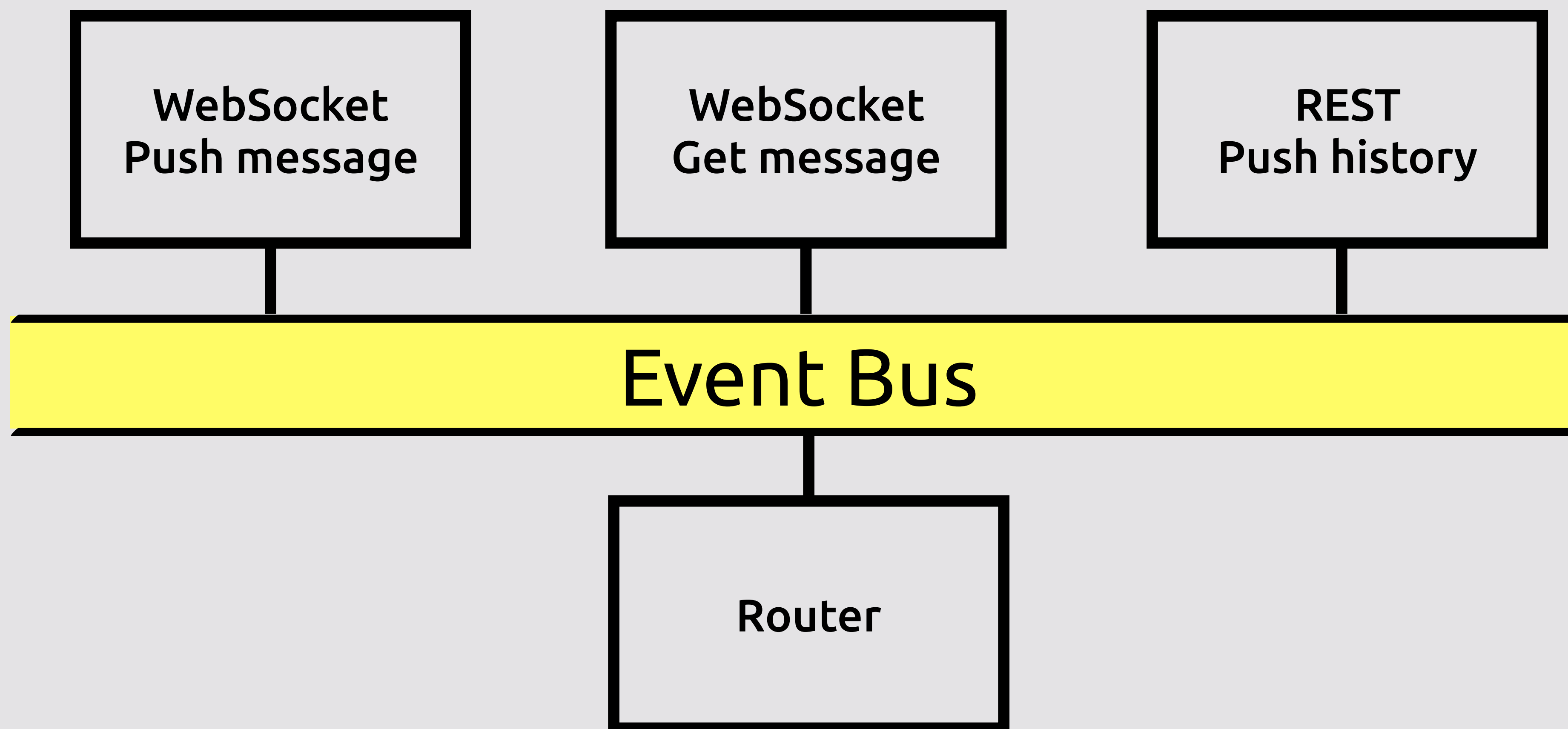


Продумывайте  
маршруты



- 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

1 / 13

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

# | Отправка сообщений по REST

2 / 13

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

# | Отправка сообщений по REST

3 / 13

```
@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

4 / 13

```
@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

5 / 13

```
@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

6 / 13

```
@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

7 / 13

```
@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

8 / 13

```
@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

9 / 13

```
@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

10 / 13

```
@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

11 / 13

```
@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

12 / 13

```
@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);
}
```

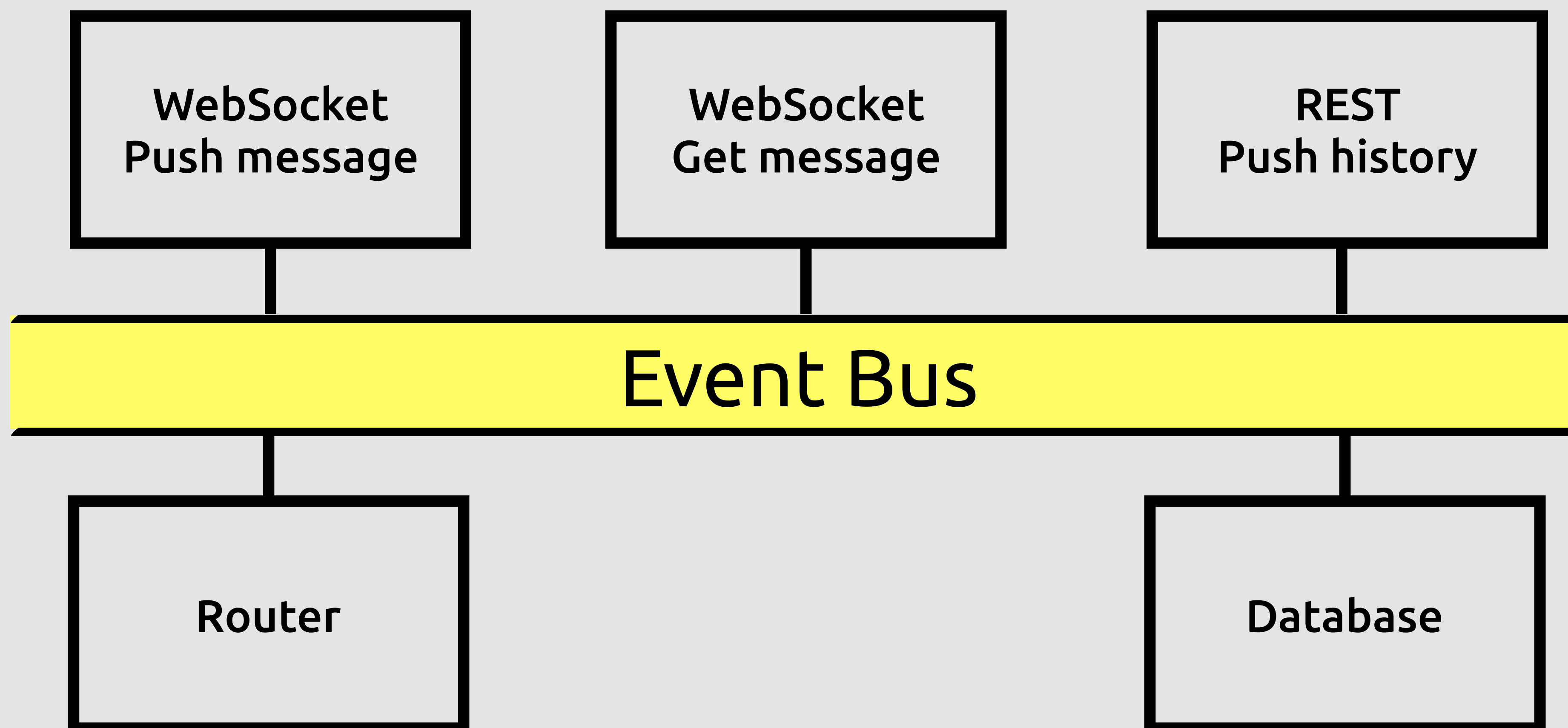
# Отправка сообщений по REST

13 / 13

Demo

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

# | Общая шина v2.2



# | MongoDB

1 / 15

```
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

2 / 15

```
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

3 / 15

```
public class MongoClient 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

4 / 15

```
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

5 / 15

```
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

6 / 15

```
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

7 / 15

```
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

8 / 15

```
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

9 / 15

```
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

10 / 15

```
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

11 / 15

```
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());  
    }  
}
```

```
void createWebSocketServer(ServerWebSocket wsServer) {  
    wsServer.frameHandler(wsFrame -> {  
  
        vertx.eventBus().publish(  
            "router", wsFrame.textData());  
        });  
    }  
}
```



# | MongoDB

13 / 15

```
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

14 / 15

```
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());  
    }  
}
```

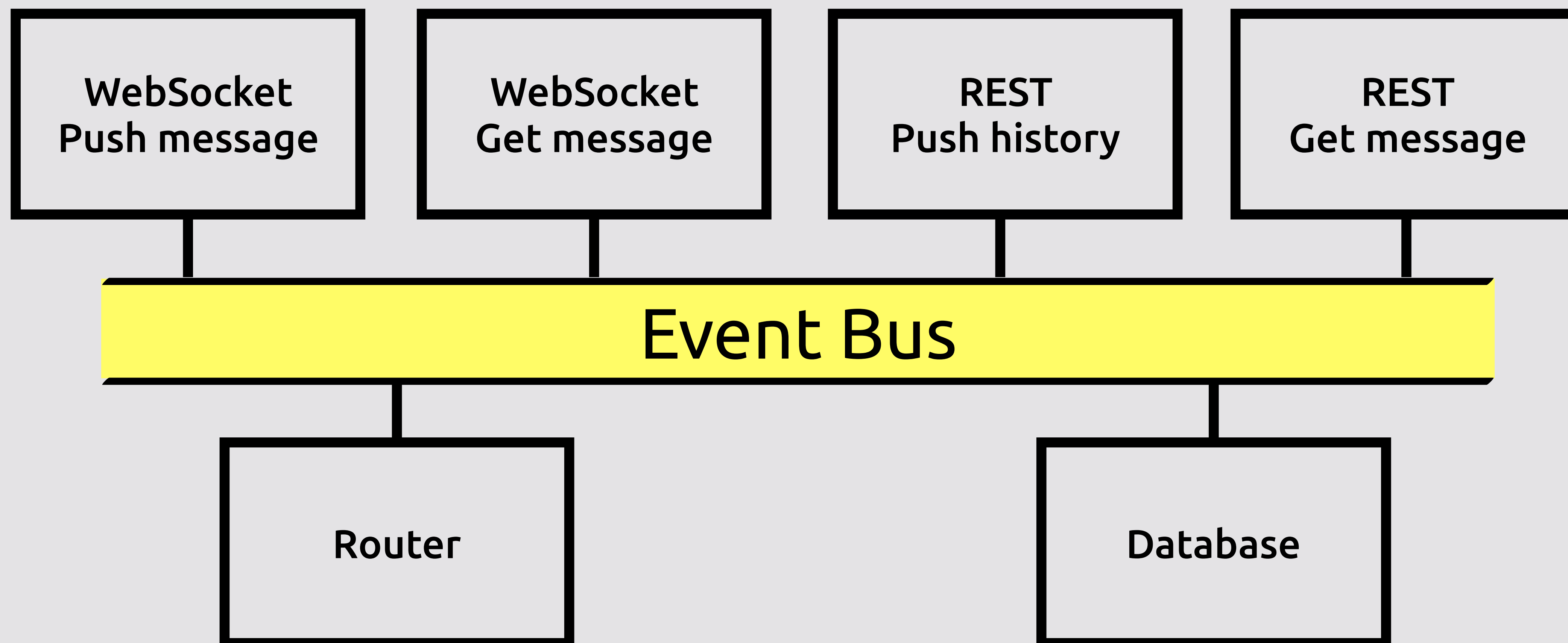
# | MongoDB

15 / 15

```
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());  
    }  
}
```

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

# | Общая шина v2.3



## | Rest-клиент

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

# | Прием сообщений по REST

1 / 12

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



# | Прием сообщений по REST

2 / 12

```
@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

3 / 12

```
@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

4 / 12

```
@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

5 / 12

```
@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

6 / 12

```
@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

7 / 12

```
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

8 / 12

```
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

9 / 12

```
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

10 / 12

```
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

11 / 12

```
@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

12 / 12

Demo

# Kubernetes описание



**kubernetes**

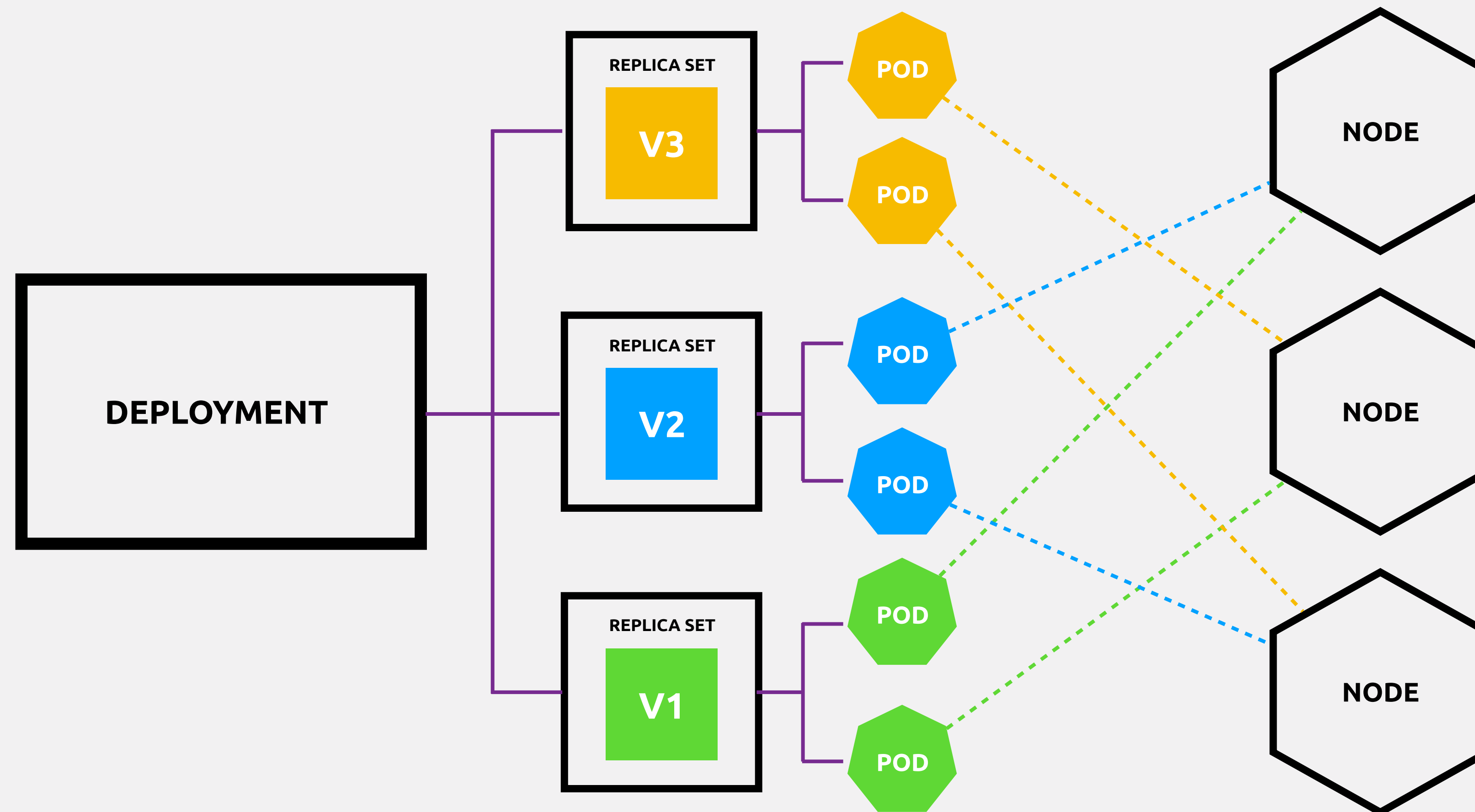
- С греческого — «Рулевой»
- Первоначальный разработчик — Google
- Управляет Docker контейнерами

# **| Kubernetes**

**Задача — автоматизировать работу с Docker контейнерами:**

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

# Что такое Deployment



# I Deploy

1 / 5

```
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());  
    }  
}
```



# I Deploy

2 / 5

```
---
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
```

# I Deploy

3 / 5

```
---
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
```

## I Deploy

4 / 5

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

# Demo

# Cluster Manager в Vert.x



# Cluster B Vert.x

1 / 11

```
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 B Vert.x

2 / 11

---

apiVersion: v1

kind: Service

metadata:

  name: jbreak-chat

  . . . .

# Cluster B Vert.x

3 / 11

```
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 B Vert.x

4 / 11

```
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 B Vert.x

5 / 11

```
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 in Vert.x

6 / 11

```
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 in Vert.x

7 / 11

```
void setupCluster() {  
    ClusterManager clusterManager =  
        new IgniteClusterManager(getIgniteConfiguration());  
  
    VertxOptions options = new VertxOptions()  
        .setClustered(true)  
        .setClusterManager(clusterManager)  
        .setClusterHost(getMyIp());  
  
    Vertx.clusteringVertx(options, res -> {  
        if (res.succeeded()) {  
            Vertx vertx = res.result();  
            deploy(vertx);  
        }  
    });  
}
```

# Cluster in Vert.x

8 / 11

```
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 in Vert.x

9 / 11

```
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 in Vert.x

10 / 11


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

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



# Demo





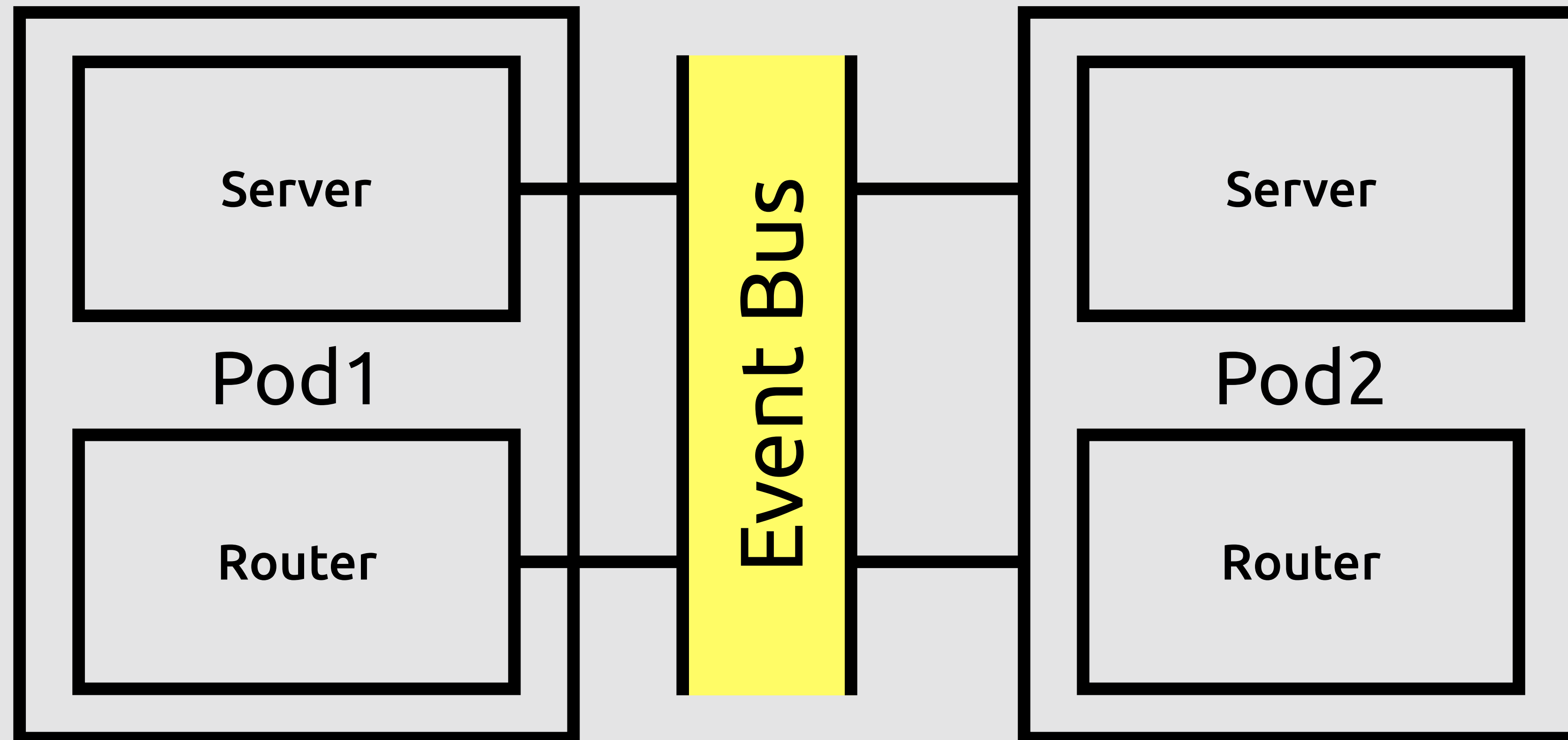
Продумывайте  
маршруты



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

# Cluster B Vert.x

1 / 15



# Cluster B Vert.x

2 / 15

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

# Cluster B Vert.x

3 / 15

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

# Cluster B Vert.x

4 / 15

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

# Cluster B Vert.x

5 / 15

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



# Cluster B Vert.x

6 / 15

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

# Cluster B Vert.x

7 / 15

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

# Cluster in Vert.x

8 / 15

```
void createWebSocketServer(ServerWebSocket wsServer) {  
    wsServer.frameHandler(wsFrame -> {  
  
        vertx.eventBus().publish(  
            "router", wsFrame.textData());  
        });  
    }  
}
```

# Cluster in Vert.x

9 / 15

```
void createWebSocketServer(ServerWebSocket wsServer) {  
    wsServer.frameHandler(wsFrame -> {  
  
        vertx.eventBus().send(  
            "router", wsFrame.textData());  
        });  
    }  
}
```

# Cluster B Vert.x

10 / 15

```
@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 B Vert.x

11 / 15

```
@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 B Vert.x

12 / 15

```
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 B Vert.x

13 / 15

```
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 B Vert.x

14 / 15

```
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

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

**ЧИСТИМ  
за собой**

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

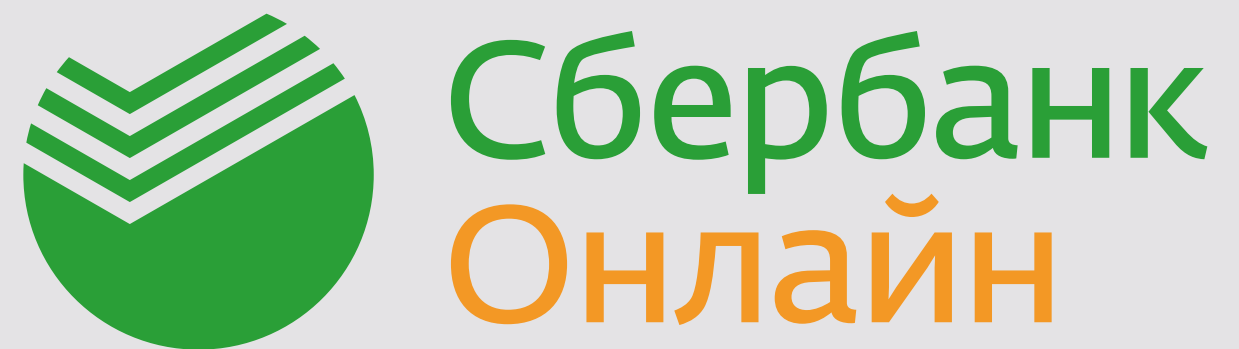
# Выученные уроки

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

The background of the image is a dense, repeating pattern of purple question marks and squares. The question marks are of various sizes and are interspersed with smaller squares, creating a textured, geometric look. The entire background is a uniform shade of purple.

Вопросы?





# Ленок Антон

Серверная разработка

Проект: Диалоги  
в Сбербанк Онлайн

[AlLenok.SBT@sberbank.ru](mailto:AlLenok.SBT@sberbank.ru)

