

Async Webapps

Vert.x, AngularJS, MongoDB

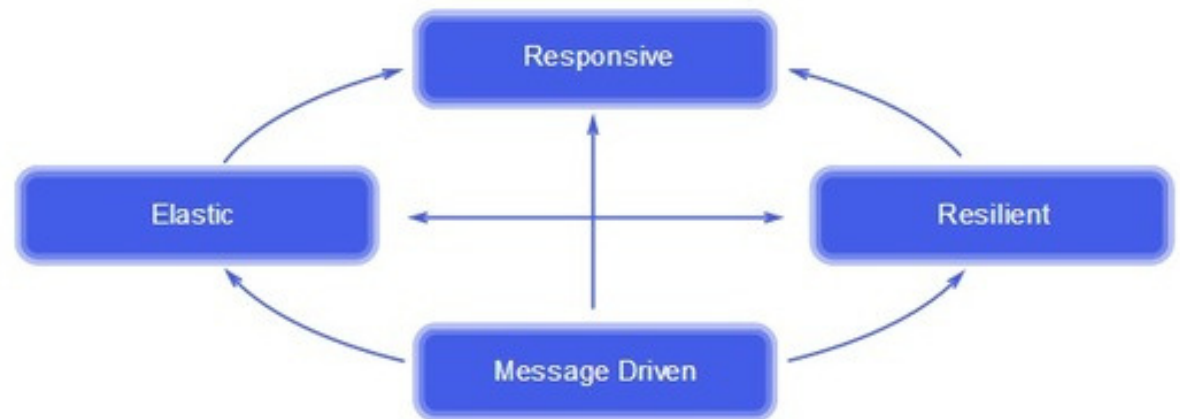
Erwin de Gier
Sogeti Java CoE
Amsterdam, Februari 2015

Demands

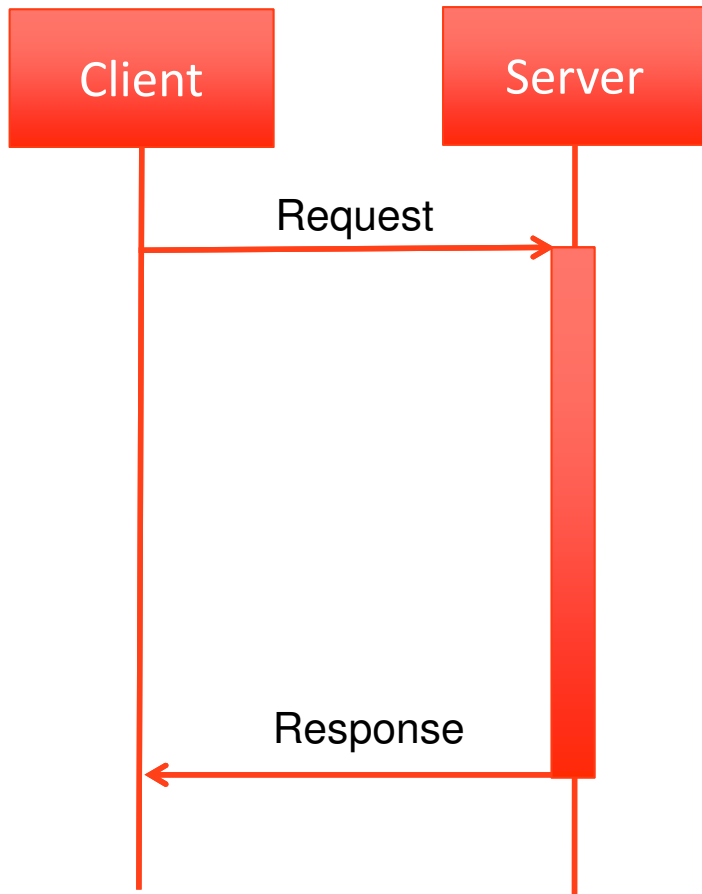
- Mobile
- Multicore
- Cloud computing
- Interactive & real-time
- Responsive
- Collaborative

Reactive manifesto

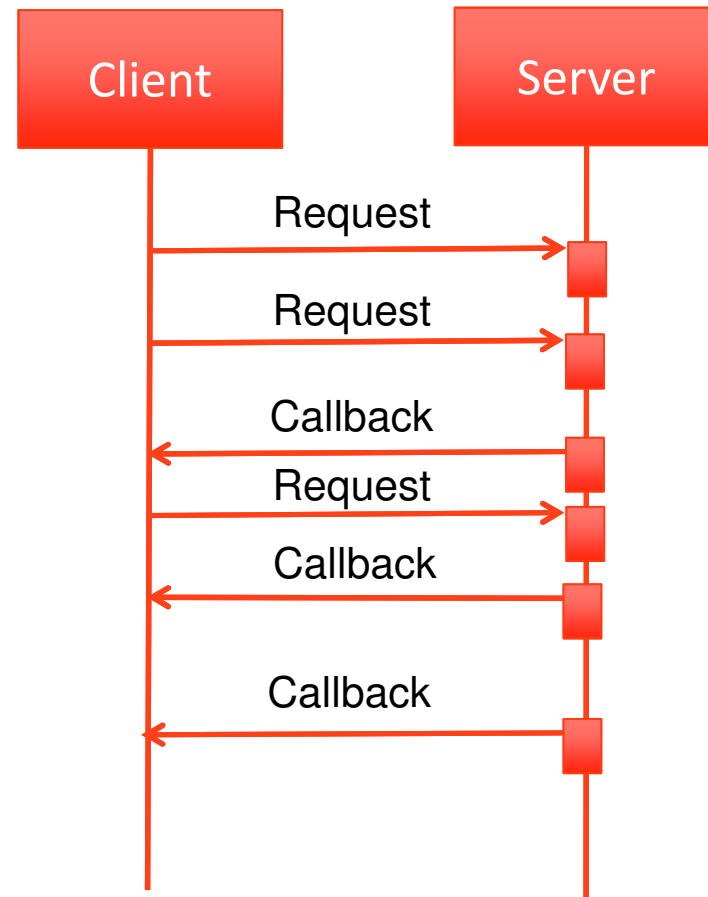
- react to events (message driven)
- react to load (scalable)
- react to failure (resilient)
- react to users (responsive)



Blocking vs non-blocking



One thread per connection (1 client)



One thread per event-loop (multiple clients)

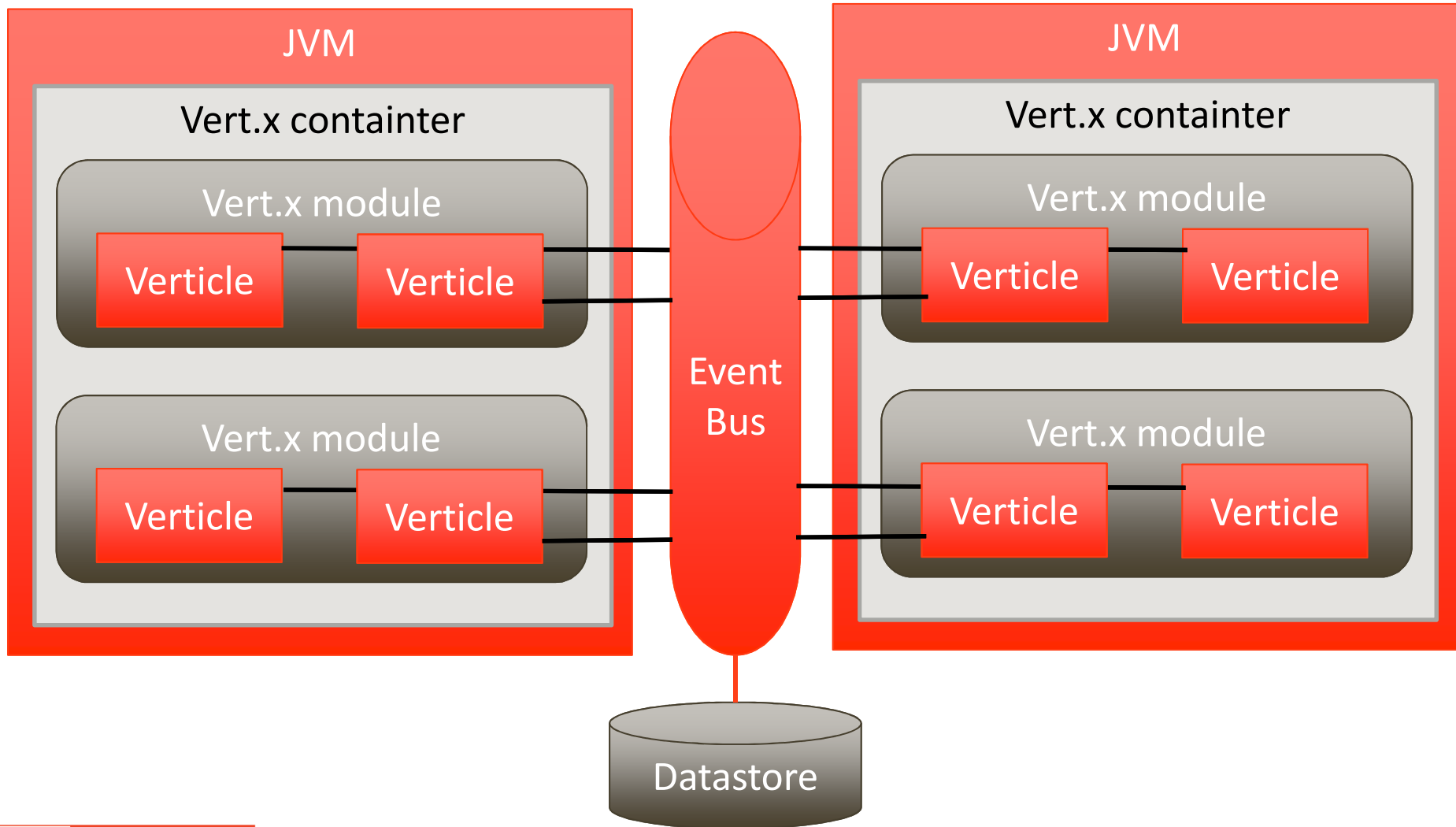
Vert.x

- c10k problem
- Polyglot, running on JVM
- Asynchronous and synchronous
- Scalable
- Distributed eventbus
- Thread per event-loop, non blocking
- Micro services

Popular Technologies

- AngularJS: Javascript MVC framework
- VertX: Asynchronous Polyglot JVM library
- MongoDB: NoSQL database

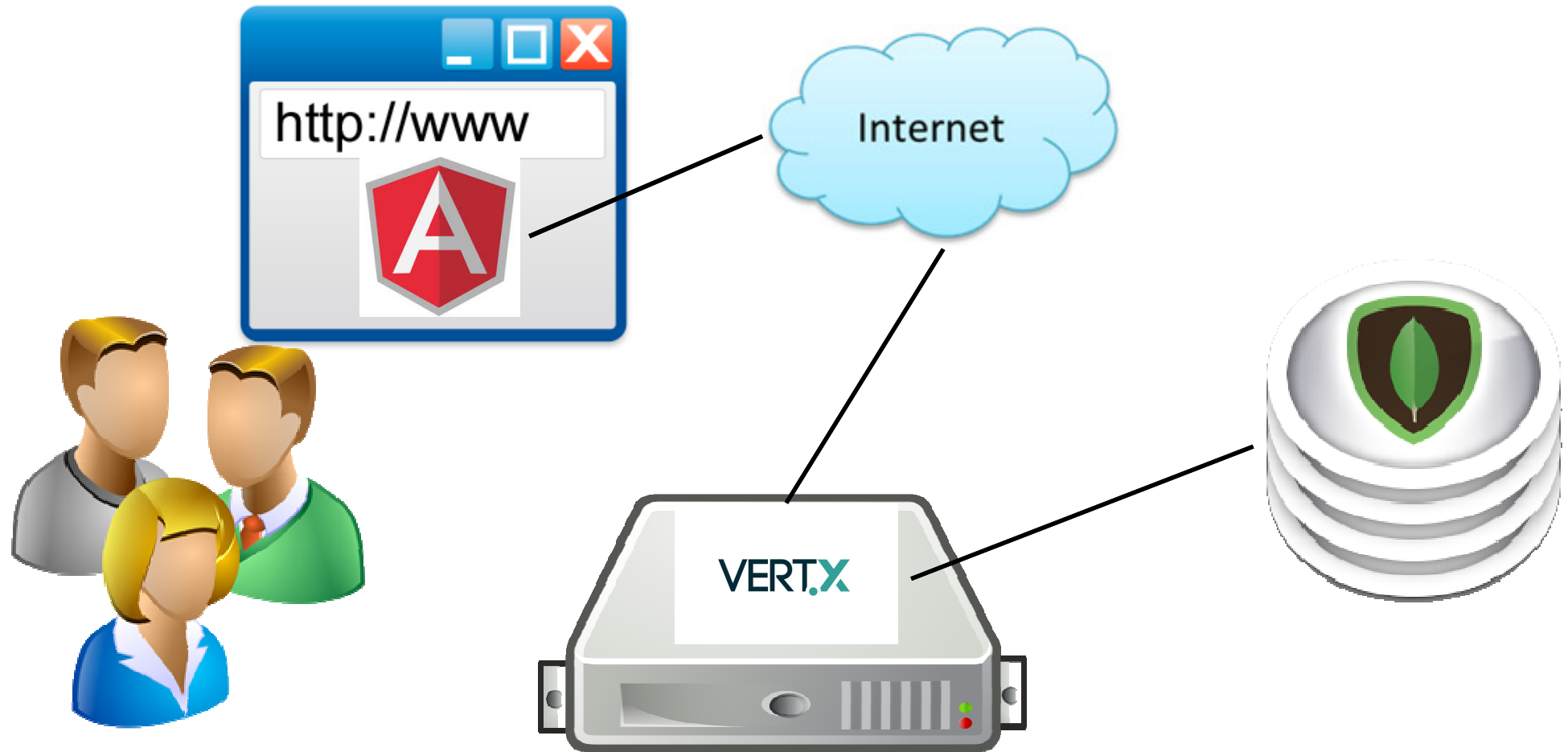
Vert.x



Chat application

- Send messages
- Receive messages
- Network communication
- Persist messages
- Find messages

Architecture



Architecture



Getting started

- Java SDK (JDK) 8
- Maven
- IDE (Eclipse)
- MongoDB
- git (optional)
- check versions:
 - `mvn -X`

Starter project

- git clone
<https://github.com/erwindeg/vertx3-chat-start.git>
- mvn clean install
- java -jar target\`<jar-name>`-fat.jar
- <http://localhost:8080>

Run it

- Eclipse:
 - run as java application
 - main class: `io.vertx.core.Starter`
 - run `nl.sogeti.MainVerticle`
- CLI:
 - `Java -jar target\<jar-name>-fat.jar`

Server MainVerticle.java

```
HttpServer server = vertx.createHttpServer(new  
HttpServerOptions().setPort(8080).requestHandler(req ->  
matcher.accept(req));
```

```
SockJSServer.sockJSServer(vertx, server).bridge(new  
SockJSServerOptions().setPrefix("/eventbus"),  
new BridgeOptions().addInboundPermitted(new  
JsonObject()).addOutboundPermitted(new JsonObject()));
```

```
server.listen();
```

Client angular main.html

```
<div ng-controller="MainCtrl">
  <select ng-model="selected" ng-options="message.text as
message.name + ': ' + message.text for message in messages
| orderBy:'date':true"
  multiple size="20" style="min-width: 90%;">
</select>
<form ng-submit="sendMessage()">
  <input type="text" ng-model="message.name"
placeholder="Type your name here" /> <input type="text" ng-
model="message.text" placeholder="Type your message here"/>

  <input type="submit" value="Send" />
</form>
</div>
```


Assignment 1

- Send messages to the vertx eventbus

```
<script>

    var eb = new vertx.EventBus('http://localhost:8080/eventbus');

    eb.onopen = function() {

        eb.registerHandler('some-address', function(message) {

            console.log('received a message: ' + JSON.stringify(message));

        });

        eb.publish('some-address', {name: 'tim', age: 587});

    }

</script>
```

Answer assignment 2

```
angular.module('resourcesApp').controller('MainCtrl',
function($scope, $resource) {
    $scope.messages = [];
    var eb = new
vertx.EventBus('http://' + window.location.host + '/eventbus');
    eb.onopen = function() {
        eb.registerHandler('chat', function(message) {
            $scope.messages.push(message);
            $scope.$apply();
        });
    }
    $scope.sendMessage = function() {
        $scope.message.date = Date.now();
        eb.publish('chat', $scope.message);
        $scope.message.text = "";
    };
});
```

Assignment 2

- Log the messages on the server

```
vertx.eventBus().consumer(String address, Handler<Message<T>  
message);
```

Answer assignment 2

```
vertx.eventBus().consumer("chat",  
message -> System.out.println(message.body()));
```

Cluster mode

- `-cluster -cluster-host <ip_address>`

Start MongoDB

- `mongo\bin\mongod.exe -dbpath data`

MongoDB persistence

```
private static final String MONGO_ADDRESS =  
    UUID.randomUUID().toString();  
MongoService mongoService;
```

```
public void start() throws Exception {  
    mongoService = setUpMongo();  
    ...
```

```
private MongoService setUpMongo() {  
    DeploymentOptions options = new  
        DeploymentOptions().setConfig(new  
            JsonObject().put("address", MONGO_ADDRESS));  
    vertx.deployVerticle(new MongoServiceVerticle(), options,  
        res -> System.out.println(res.result()));  
    return MongoService.createEventBusProxy(vertx,  
        MONGO_ADDRESS);  
}
```

Assignment 3

- Save the messages to MongoDB

```
JsonObject document = new JsonObject().put("title", "The Hobbit");  
mongoService.insert("books", document, res -> {  
  
    if (res.succeeded()) {  
  
        String id = res.result();  
        System.out.println("Inserted book with id " + id);  
  
    } else {  
        res.cause().printStackTrace();  
    }  
  
});
```


Answer assignment 3

```
public void start() throws Exception {  
    ...  
    vertx.eventBus().consumer("chat", this::saveMessages);  
    ...  
  
    private void saveMessages(Message message) {  
        proxy.insert("messages", new  
            JsonObject(message.body().toString()), res ->  
            System.out.println(res.succeeded()));  
    }  
}
```

Assignment 4

- Return all saved messages via REST

```
JsonObject query = new JsonObject();

mongoService.find("books", query, res -> {

    if (res.succeeded()) {

        for (JsonObject json : res.result()) {

            System.out.println(json.encodePrettily());

        }

    }

}
```



Answer assignment 4

```
$scope.messages = $resource('/api/history').query();
```

```
matcher.matchMethod(RequestMethod.GET, "/api/history", req ->  
proxy.find("messages", new JsonObject(), res ->  
req.response().end(new JSONArray(res.result()).toString())));
```

Unit Testing

```
@RunWith(VertxUnitRunner.class)
public class MyJUnitTest {

    private Vertx vertx;

    @Before
    public void setUp(TestContext context) {
        Async async = context.async();
        vertx = Vertx.vertx();
        vertx.deployVerticle(MainVerticle.class.getName(), ar -> {
            if (ar.succeeded()) {
                async.complete();
            } else {
                context.fail("Could not deploy verticle");
            }
        });
    }
}
```

Unit Testing

```
@Test
public void testHello(TestContext context) {
    Async async = context.async();
    HttpClient client = vertx.createHttpClient();
    HttpClientRequest req = client.get(8080, "localhost", "/app/test.html");
    req.exceptionHandler(err -> {
        context.fail();
    });
    req.handler(resp -> {
        context.assertEquals(200, resp.statusCode());
        Buffer entity = Buffer.buffer();
        resp.handler(entity::appendBuffer);
        resp.endHandler(v -> {
            context.assertEquals("test", entity.toString("UTF-8"));
            async.complete();
        });
    });
    req.end();
}
```



Unit Testing

@After

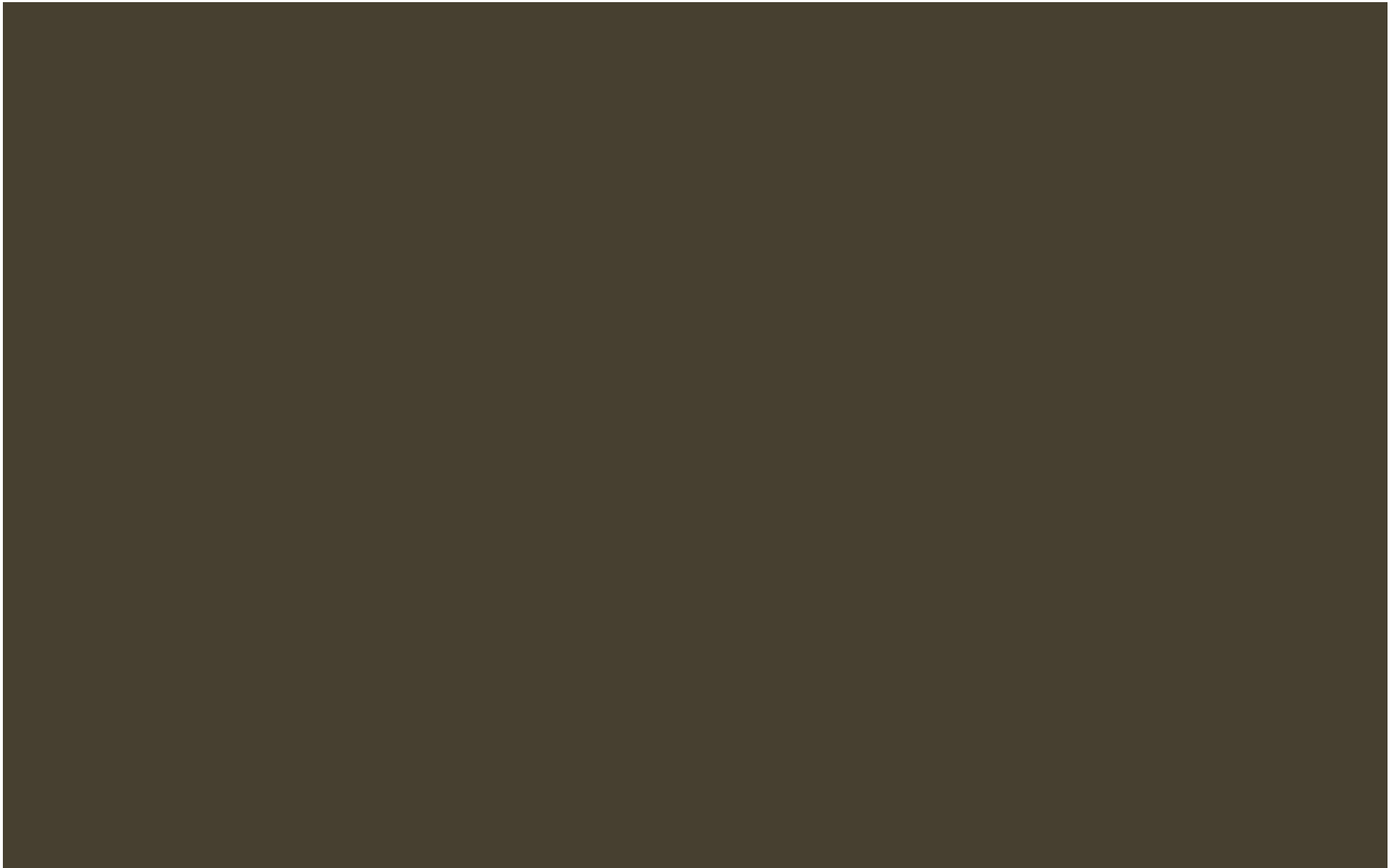
```
public void tearDown(TestContext context) {  
    Async async = context.async();  
    vertx.close(ar -> {  
        async.complete();  
    });  
}
```

Assignment 5

- Test sending a message on the eventbus
 - See `MyUnitTest.java`

Answer assignment 5

```
@Test
public void testSendMessage(TestContext context) {
    Async async = context.async();
    vertx.eventBus().consumer("chat", message -> {
        context.assertEquals("test message",
            ((JsonObject) message.body()).getString("message"));
        async.complete();
    });
    vertx.eventBus().publish("chat",
        new JsonObject("{\"message\": \"test message\"}"));
}
```

Mongo DB unique index

- `mongo\bin\mongo.exe`
- use `default_db`
- `db.messages.createIndex({ date: 1, name: 1, text: 1 }, { unique : true })`

Receive messages

```
private final String channel = UUID.randomUUID().toString();

public void start() throws Exception {
    ...
    vertx.eventBus().consumer(this.channel, this::saveMessages);
    ...
}

private void sendHistoryRequest(AsyncResult<String> result){
    vertx.eventBus().publish("history", new
    JsonObject().put("channel", this.channel));
}
```

Send messages

```
vertx.eventBus().consumer("history", m -> proxy.find("messages",  
new JsonObject(), res -> sendMessages(((JsonObject)  
m.body()).getString("channel"), res)));
```

```
private void sendMessages(String channel,  
    AsyncResult<List<JsonObject>> result) {  
    if(!this.channel.equals(channel)){  
        for (JsonObject message : result.result()) {  
            System.out.println("sending message: "+message);  
            vertx.eventBus().send(channel, message);  
        }  
    }  
}
```

Send request for messages

```
private MongoService setUpMongo() {  
    ...  
    vertx.deployVerticle(new MongoServiceVerticle(), options,  
        this::sendHistoryRequest);  
    ...  
}
```