BackEnd Test Introduction

QA Automation Certification - Week 2

Agenda

- Web Services
- Client - Server Model
- HTTP - HTTPS
- Request - Response
- SOAP vs REST

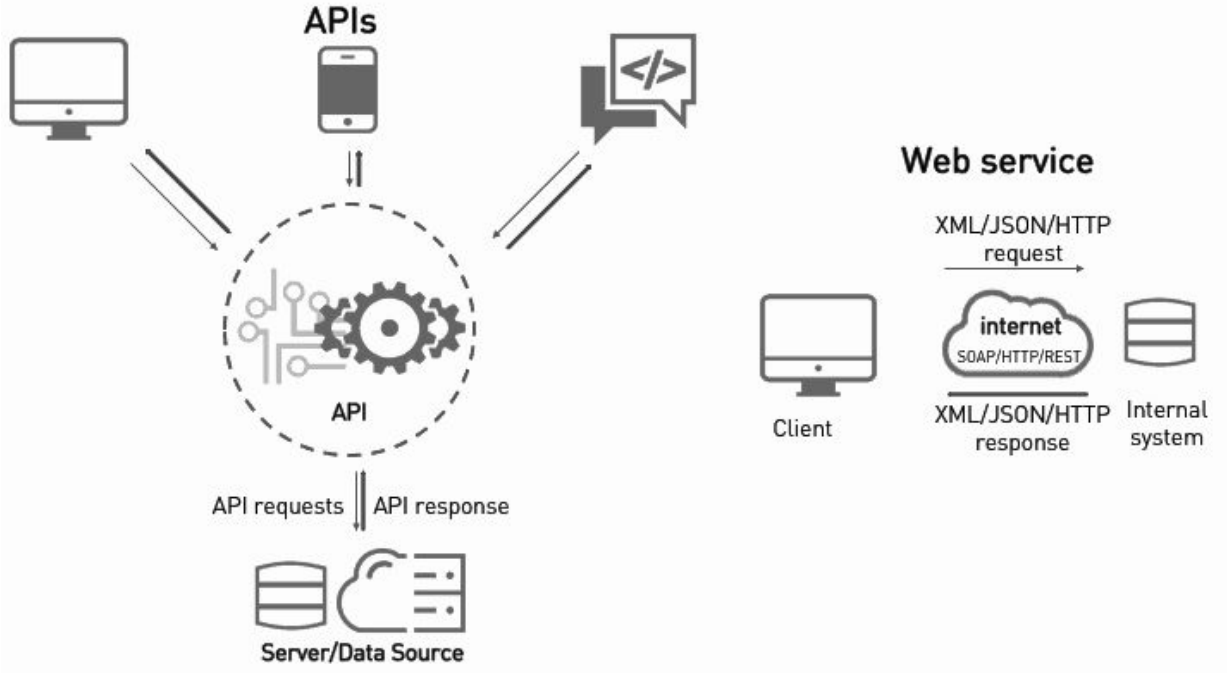
WIZELINE

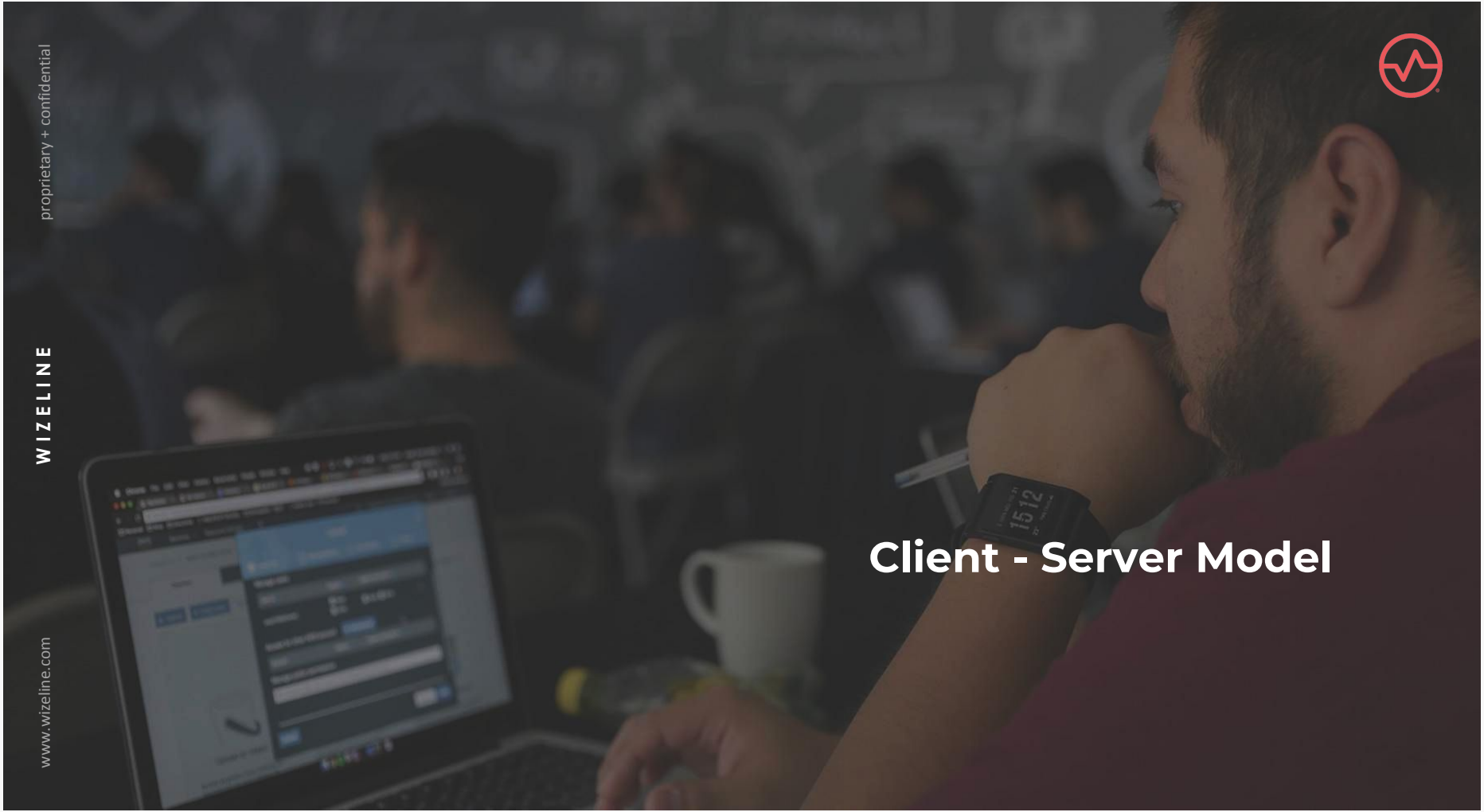
www.wizeline.com

Web Services

API - Web Service

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network.




proprietary + confidential
WIZELINE
www.wizeline.com
Client - Server Model

CLIENT

- Agent who wants to consume a specific information from an external source.
- Request a server's content or a service function.

SERVER

- Provider of a resource.
- Receives requests for information from multiple clients and provides a response only if the protocol conditions are respected.

WIZELINE

What is a Protocol?

Special set of rules that end points in a telecommunication connection use when they communicate/interact.

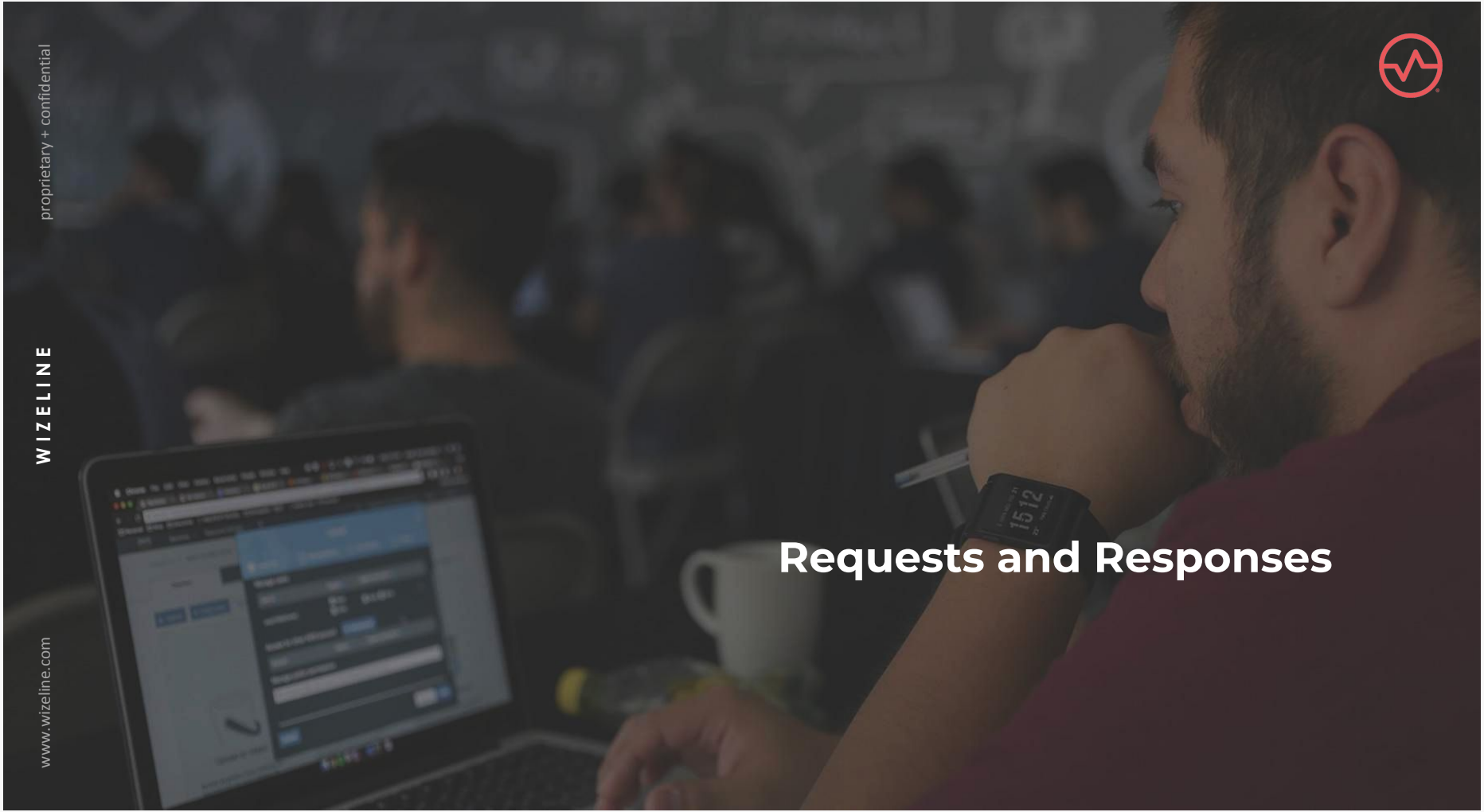
Protocols specify interactions between the communication entities.

HTTP

- Hypertext Transfer Protocol.
- It is the foundation of data communication for the World Wide Web.
- Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text.
- HTTP is the protocol to exchange or transfer hypertext.

HTTPS

- Hypertext Transfer Protocol Secure.
- Secure version of HTTP.
- The communication between your browser and the website are encrypted.
- Used to protect highly confidential online transactions like online banking, shopping, etc.


proprietary + confidential
WIZELINE
www.wizeline.com
Requests and Responses

HTTP Request and Responses

1. Browser sends a request sent to the web server that hosts the website.
2. The web server then returns a response as a HTML page or any other document format to the browser.
3. Browser displays the response from the server to the user.

Request

- **Opening Line**
  - POST *https://beta.todoist.com/API/v8/projects*
- **Header Lines**
  - Authorization: Bearer $token
  - Content-Type: application/json
- **Message Body**
  - {

    "name": "New Project"

    }

Response

- **Status Line**
  - 200 OK
- **Headers**
  - Content-Type: application/json
  - Content-Length: 94
- **Message Body**
  - {

    "id": 2192924505,

    "name": "New Project",

    "order": 6,

    "indent": 1,

    "comment_count": 0

    }

Common Response Status Codes

HTTP Status Codes

Level 200 (Success)

200 : OK

201 : Created

203 : Non-Authoritative Information

204 : No Content

Level 400

400 : Bad Request

401 : Unauthorized

403 : Forbidden

404 : Not Found

409 : Conflict

Level 500

500 : Internal Server Error

503 : Service Unavailable

501 : Not Implemented

504 : Gateway Timeout

599 : Network timeout

502 : Bad Gateway

CRUD

Create	Read	Update	Delete
• POST • Creates new resources	• GET • Retrieve existing resources	• PUT • Update existing resources with new information	• DELETE • Removes existing resources

WIZELINE

SOAP

- XML-based message protocol
- Uses WSDL for communication between consumer and provider
- Invokes services by calling RPC method
- Doesn’t return human readable result
- Transfer is over HTTP, SMTP, FTP, etc.
- JavaScript can call SOAP, but it is difficult to implement
- Performance is not great compared to REST

REST

- Architectural style
- Uses XML or JSON to send and receive data
- Simply calls services via URL path
- Result is readable which is just plain XML or JSON
- Transfer is over HTTP only
- Easy to call from JavaScript
- Performance is much better compared to SOAP - less CPU intensive, etc.

Agenda

- API Testing
- Tools
- Postman
- BDD
- Newman

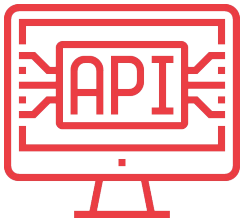
API TESTING

WIZELINE

www.wizeline.com

API Testing Intro

- API Testing creates a more reliable code helping the process of development, testing, validation, and documentation.
- Automation helps with the coverage of positive, negative, and edge scenarios.
- Nothing is left to chance and that all parameters and permutations are tested.



WIZELINE

Tools

Rest Assured

- Java based scripting.
- Integrated with build tools (Maven, gradle, etc.)
- BDD (Cucumber, JBehave, etc.).

Validate Status Code of 200 using Rest Assured:

```
@Test public void
lotto_resource_returns_200_with_expected_id_and_winn
ers() {

    when().
            get("/lotto/{id}", 5).
    then().
            statusCode(200);
```

Tools

Frisby.js

- JavaScript based.

Validate Status Code of 200 using Frisby.js:

```
const frisby = require('frisby');

it ('should return a status of 200', function ()
{
  return frisby
    .get('http://api.example.com')
    .expect('status', 200);
});
```

Tools

Postman

- JavaScript based scripting.
- User friendly.
- Newman CLI for CI integration.
- Postman BDD.
- Independent UI

Validate Status Code of 200 using Postman:

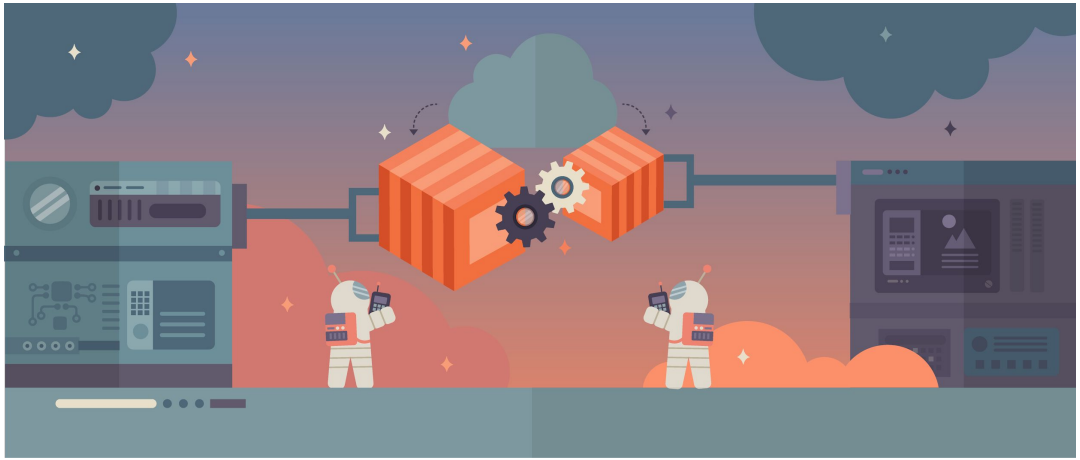
```
pm.test("Status code is 200", () => {
    pm.response.to.have.status(200);
});
```

WIZELINE

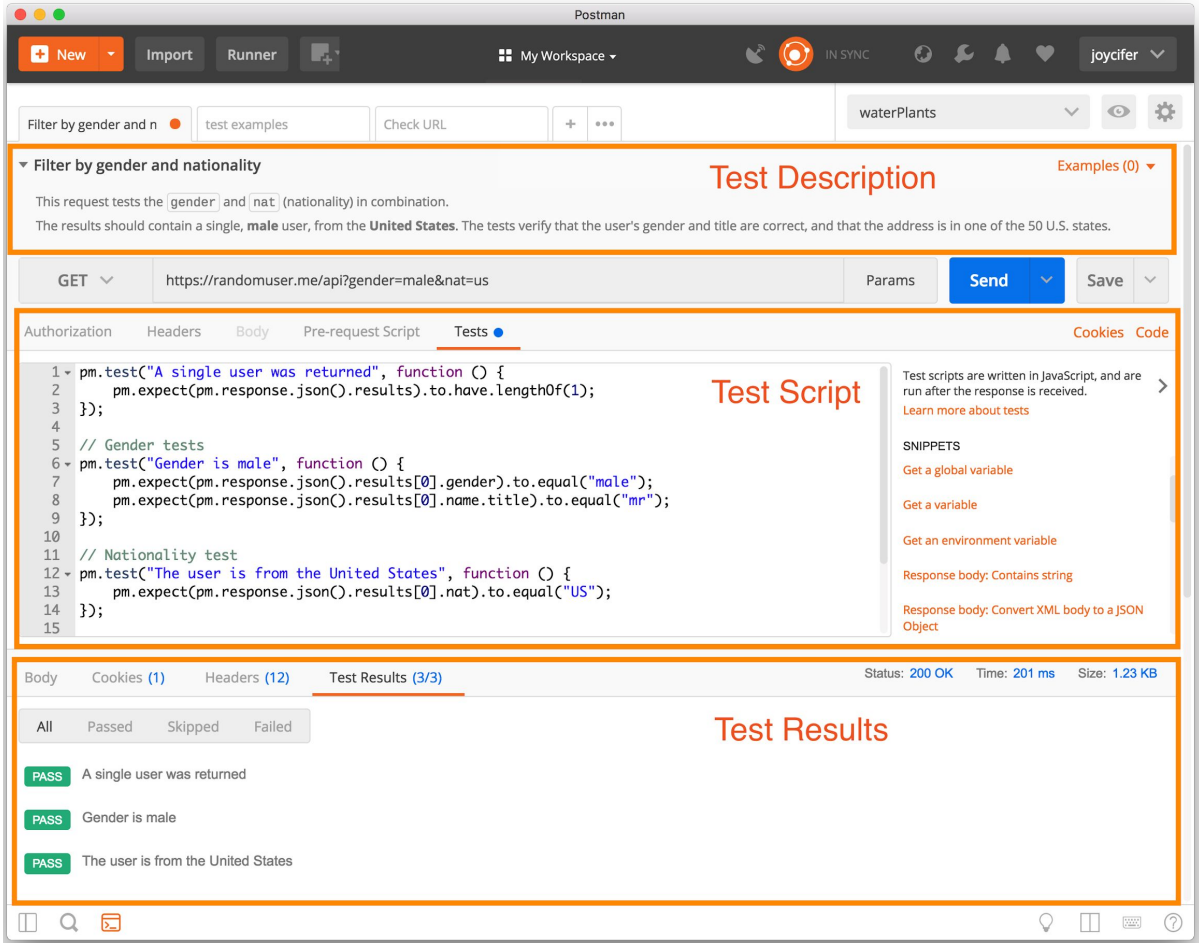
POSTMAN

What is a Postman?

- Web Development and Testing tool for APIs.
- Available for Windows, MacOs, and Linux.
- Browser plugin no longer supported nor updated.
- GraphQL Support!



Postman Application



What is BDD?

Business readable and domain-specific language that allows you to describe a system's behavior without explaining how that behavior is implemented.

Communication | **Specifications** | **Cucumber** | **Gherkin**

- Business readable language
- Reviewed by stakeholders
- Reviewed by tech team
- Easy agreement on what and how to test

- Specification-based description
- Scenarios typically written before anything else
- Easy to understand for the entire team

- Java Based
- Features
- Gherkin language

- **Given** [initial context],
- **When** [event occurs],
- **Then** [assert outcome]

Postman BDD

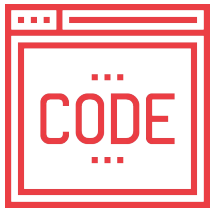
Simple Language

The language is understandable by all the members of the team, which reduces misconceptions and misunderstanding and makes it easier for new members to join the working process.

Strong collaboration

BDD is meant to be collaborative. Developers and testers can work as a team to write the tests.

Chai Mocha



Chai is an *assertion library*, similar to Node's built-in assert. It makes testing much easier by giving you lots of assertions you can run against your code.

Installation

Hit the following end point:

Get *http://bigstickcarpet.com/postman-bdd/dist/postman-bdd.min.js*

Pre-req

```
if (!environment.postman_bdd_path) {

    postman.setGlobalVariable('postman_bdd_path',
'http://bigstickcarpet.com/postman-bdd/dist/postman-bdd.min.js');

}
```

Test

```
postman.setGlobalVariable('postmanBDD', responseBody);
```

Assertions Styles

Assert Example

```
assert.notEqual(3,4 ‘These numbers are not equal’);
```

Expect Example

```
expect(‘hello’).to.equal(‘hello’);
```

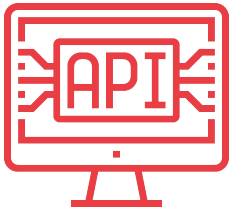
Should Example

```
 it('Should return a 200 response', () => {
      response.should.have.status(200);
    });
```

WIZELINE

Newman

What is Newman?



Newman is a command-line collection runner for Postman. It allows to run a Postman collection directly from the command-line. Can be easily integrated with a CI and build systems.

Installation and Usage

Newman is built on *Node.js* so you need to install node before, then run:

```
npm install -g newman
```

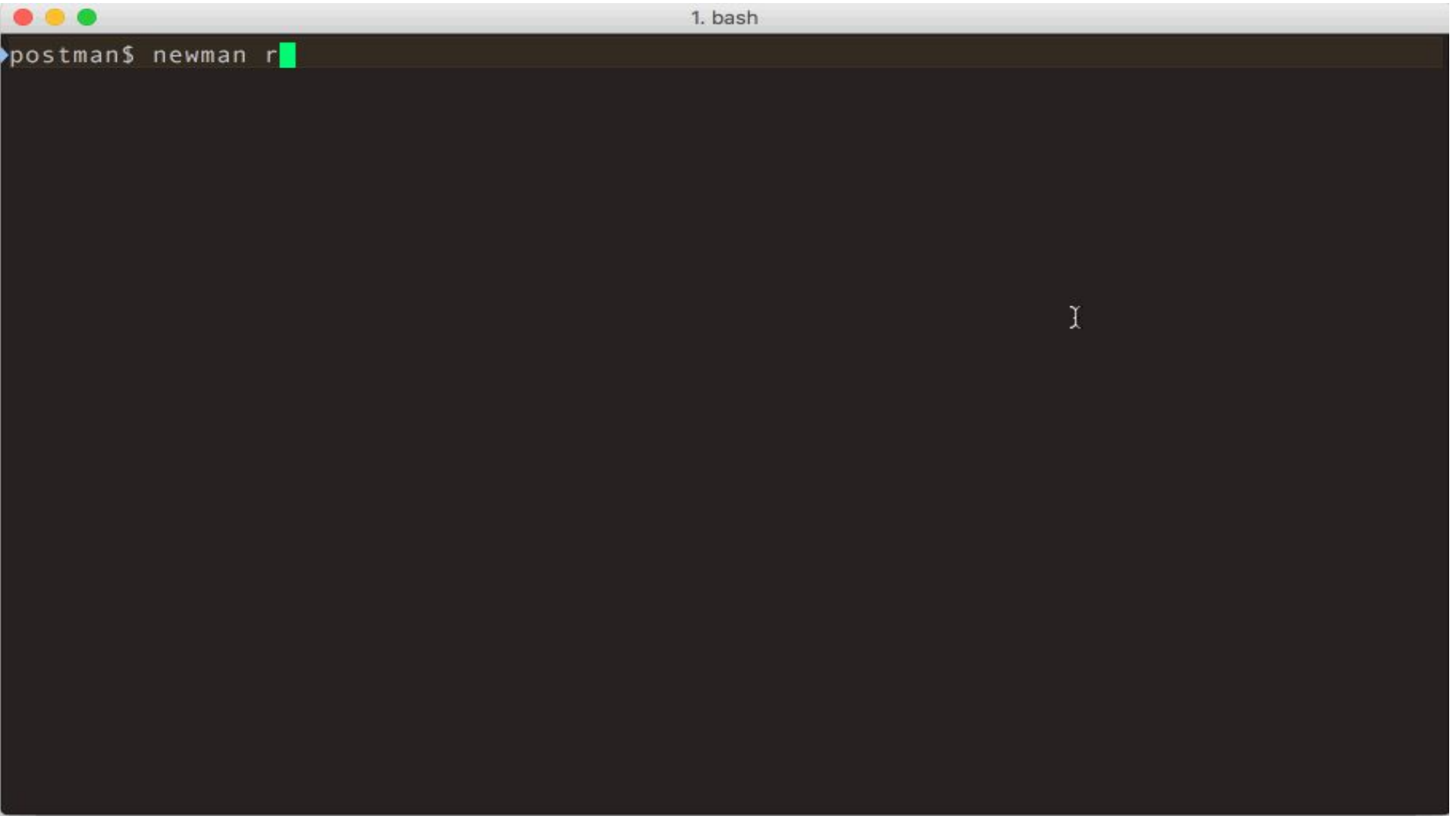
Then you can easily run any existing collection with the follow command:

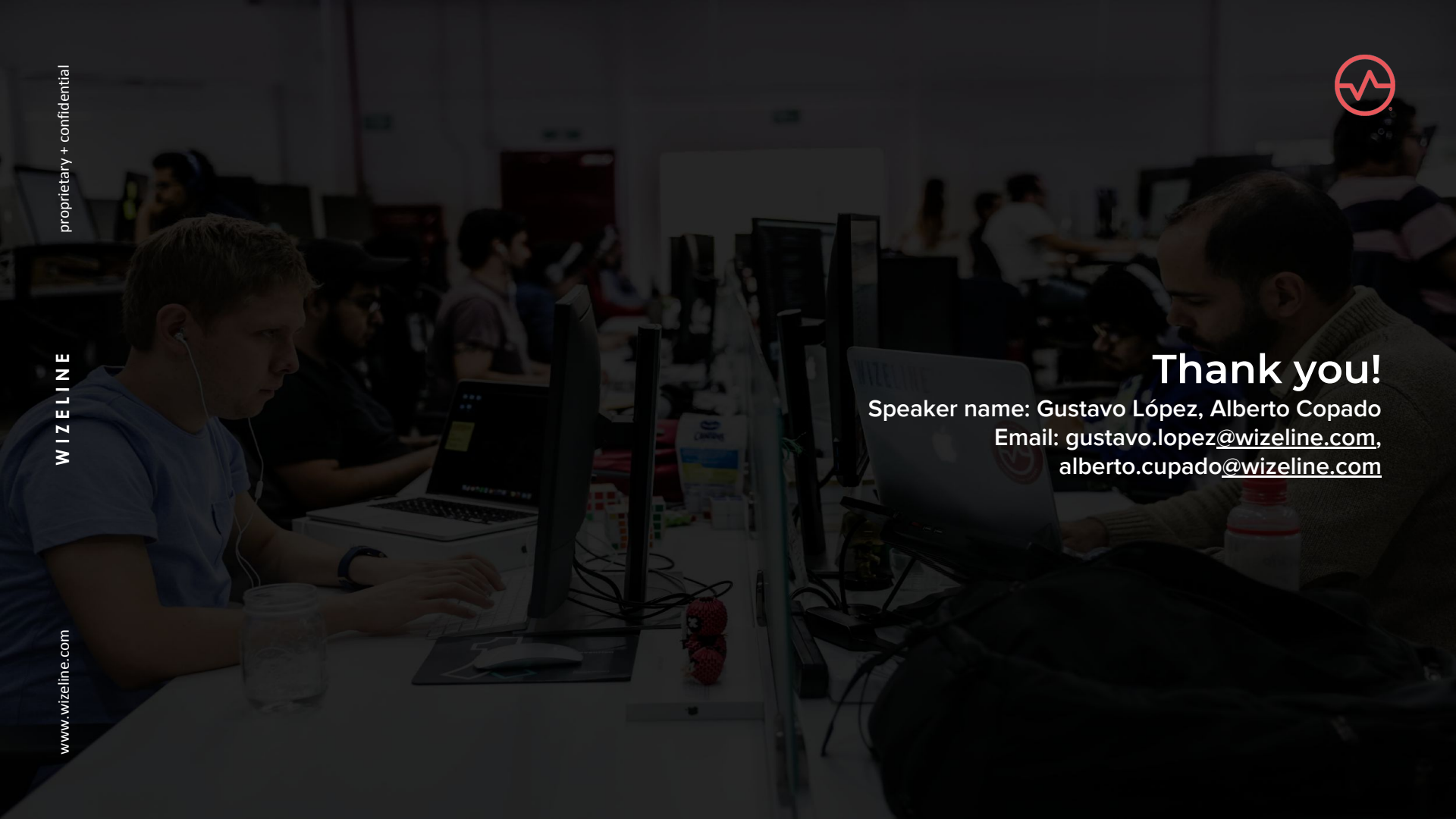
```
npm run SampleCollection.json
```

There are options to customize a run with the following parameters:

```
Options:
-h, --help                          Output usage information
-v, --version                       Output the version number
Basic setup:
--folder [folderName]               Specify a single folder to run from a collection.
-e, --environment [file|URL]        Specify a Postman environment as a JSON [file]
-d, --data [file]                   Specify a data file to use either json or csv
-g, --globals [file]                Specify a Postman globals file as JSON [file]
-n, --iteration-count [number]      Define the number of iterations to run
Request options:
--delay-request [number]            Specify a delay (in ms) between requests [number]
--timeout-request [number]          Specify a request timeout (in ms) for a request
Misc.:
--bail                              Stops the runner when a test case fails
--silent                            Disable terminal output
--no-color                          Disable colored output
-k, --insecure                      Disable strict ssl
-x, --suppress-exit-code            Continue running tests even after a failure, but
                                    exit with code=0
--ignore-redirects                  Disable automatic following of 3XX responses
```

RUNNING EXAMPLE


1. bash
postman$ newman r



proprietary + confidential

WIZELINE

www.wizeline.com

Thank you!

Speaker name: Gustavo López, Alberto Copado
Email: gustavo.lopez@wizeline.com,
alberto.cupado@wizeline.com