Guide for delivering the Backend part for the Hypermedia project

June 14, 2019

Contents

1 Version history

Date	Tag	Note
2019-03-05	v.2019.0	Initial version
2019-05-24	v.2019.1	Add FAQ section
2019-05-27	v.2019.2	Improve FAQ section
2019-06-10	v.2019.3	Clarify some requirements
2019-06-14	v.2019.4	Extend 2.1 to clarify delivery of D3

2 Introduction

This guide will help you through the delivery of the backend part of your hypermedia project. If you feel that anything in this guide is missing, please contact vittorio.zaccaria@polimi.it. First of all, let us introduce the expected deliverables, i.e., the artifacts on which you will be graded. Please note that for any technical issue concerning your project you must use the template provided in this guide.

2.1 Deliverables

- D0: the web application accessible over a public address (from here on, we will use `ADDR` to identify this address) on the internet. It is suggested to use a cloud platform such as Heroku for hosting the application but other platforms (Azure, AWS etc..) are acceptable.

- D1: a main documentation page available online at url `ADDR/backend/main.html`. You will find a template for this documentation page at the end of this guide. This page ***must contain links to D2-D5*** plus other information that we will describe successively.
- D2: the OpenAPI specification of your Backend (in the form of a YAML file) available at `ADDR/backend/spec.yaml`.
- D3: a SwaggerUI page available online at `ADDR/backend/swaggerui`. This page must be generated directly from the OpenAPI specification D2 and must provide ***working*** tests against the web application API. A working test must have example data (specified in D2) that can be used to generate the request with the `Try it out button` (or equivalent).

  For the `login` request, the example data must be a valid user/password pair so that, after its execution, requests to user-private data (e.g., carts, personal info and so on) are allowed. Any deviation from this pattern must be documented in D1.

  To avoid any problem after delivery, please check yourself that tests at `ADDR/backend/swaggerui` do work.
- D4: the source code of D0 as a zip file available at `ADDR/backend/app.zip` and
- D5: the address of the online source control repository containing D4 (Github or Bitbucket). Note that the repository ***must be private***. You are not required straight away to give access to the instructor but it might be that you are asked to do so in order for the instructor to check the commit history.

2.2 How to deliver

Just before the exam date, each team will be asked to fill a form where all the pointers to deliverables D0 and D1 must be provided. The instructor will follow the link to D2-D5 that you have provided in D1.

2.3 Grading rubric

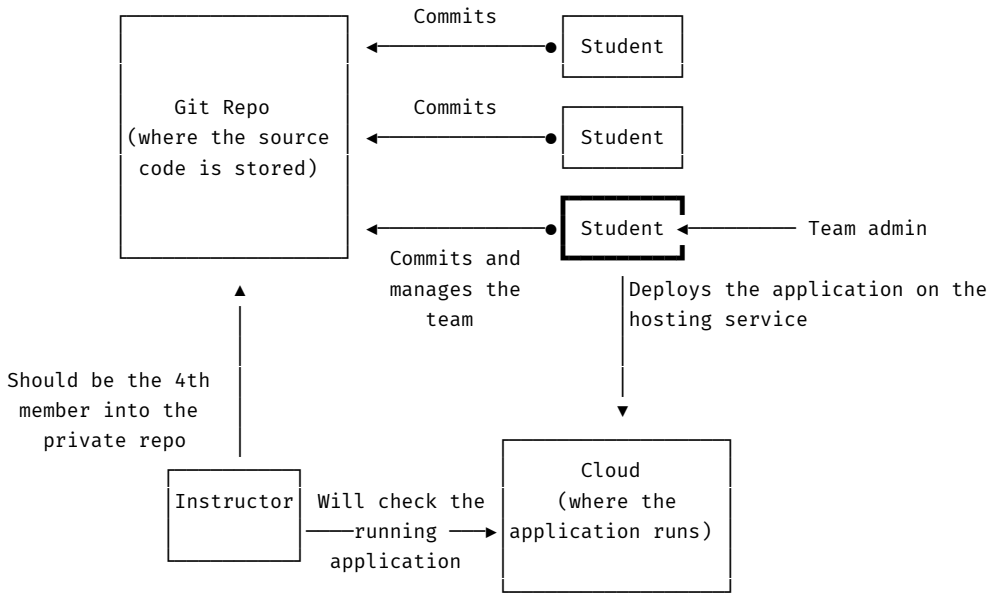
The grading rubric is at the end of this guide. Please note that: To pass the back-end part, each criteria grade should at least be ”partially” or more.

3 Appendix

3.1 Guidelines on managing your code with a Git repository

Every project should host its code on a ***private*** Git repository and provide a running application on cloud hosting platform. As far as this guide is concerned, we will distinguish the following actors:

- Instructor (V. Zaccaria/M. Gianotti)
- Team (composed of max. 3 students, where one of them will be elected as *team administrator*)



All the team members are required to commit and push their work to the common team repository under their individual name so that instructors will be able to discern the actual contribution ratio for each of them. The team manager (one of the three students, perhaps the most tinkerer) is responsible for

- management of the Bitbucket repository (e.g., tagging the final release, team member management and so on..), and
- pushing the application into production on the hosting service (Heroku).

1. Taking acquaintance with git

   We strongly urge you to ***take acquaintance with git*** by reading thoroughly the following links:

   - Learn Git with Bitbucket Cloud
   - Beginner / What is version control
   - Beginner / What is git
   - Collaborating / Syncing. Learn the basics of git ***remote***, ***fetch***, ***pull*** and ***push***.
   - Collaborating / Centralized workflow. Among the available ones, we suggest this workflow for your team as it is very simple. The suggested centralized workflow dictates every developer to have his own local copy of the entire project. To work locally on the project you could use either the command line `git` (every OS has its own way of installing it), or a graphical client. For example, Bitbucket's company has its own free application for Mac and Windows (`https://www.sourcetreeapp.com/`), feel free to check it out!.

2. IMPORTANT What should I commit over to git? When working with a Node.js based project, a common mistake done by everyone is to include in the repository the `node_modules` directory. This directory ***should not be stored in git***! This post should clarify how to avoid this mistake.

3.2 Frequently asked questions

1. On Windows, I get an `Unable to acquire a connection` error

   This happens because the default `DATABASE_URL` contains the wrong username (check this link for local setups of postgres). It has been reported that on Windows the installer uses a default username which is `postgres` so your database URL should look something like: `postgres://postgres@localhost:5432`

2. Swagger 2.X: when using the SwaggerUI, cookies do not seem to work.

   To solve this problem, note that default page serving SwaggerUI (see this link) initializes the user interface with an object:

   ```
   window.swaggerUi = new SwaggerUi({
      url: url,
      dom_id: "swagger-user interface-container", ...
   ```

   To make cookies work, the object needs an additional `enableCookies: true` flag. If you use this default page, to solve this problem you can copy the UI's `index.html`[1] and modify it accordingly by adding the missing flag:

   ```
   window.swaggerUi = new SwaggerUi({
      url: url,
      enableCookies: true,
      dom_id: "swagger-user interface-container", ...
   ```

   You must then serve explicitly this file somewhere from your assets and redirect `ADDR/backend/swaggerui` appropriately. Note that the above depends on the specific version of the installed `swagger-tools`. Some of you could also find (in place of the above):

   ```
   const ui = SwaggerUIBundle({
       url: '/backend/spec.yaml',
       dom_id: '#swagger-ui'
       ...
   })
   ```

   to which you must still add `enableCookies`:

   ```
   const ui = SwaggerUIBundle({
       url: '/backend/spec.yaml',
       dom_id: '#swagger-ui',
       enableCookies: true,
       ...
   })
   ```

4 Templates to use

4.1 Template for the on-online documentation D1

The following is a markdown file that you can convert to HTML with any tool (e.g., pandoc). Please follow the prompts to fill up coherently this template.

```
# Documentation of the Backend part

> Deliverable D1

## General group information

| Member n. | Role          | First name | Last Name | Matricola | Email address   |
| --------- | ------------- | ---------- | --------- | --------- | --------------- |
| 1         | administrator | Foo        | FooL      | 123456    | foo@example.com |
| 2         | member        | Bar        | BarL      | 78982     | bar@example.com |
| ......    |               |            |           |           |                 |
```

[1] a copy of this file is typically located at `node_modules/swagger-tools/middleware/swagger-ui/index.html` relative to your project directory.

```
## Links to other deliverables

- Deliverable D0: the web application is accessible at
  [this address](https://example.com).

- Deliverable D2: the YAML or JSON file containing the specification of the app
  API can be found at [this address](https://example.com/backend/spec.yaml).

- Deliverable D3: the SwaggerUI page of the same API is available at
  [this address](https://example.com/backend/swaggerui).

- Deliverable D4: the source code of D0 is available as a zip file at
  [this address](https://example.com/backend/app.zip).

- Deliverable D5: the address of the online source control repository is
  available [this address](https://examplegit.com). We hereby declare that this
  is a private repository and, upon request, we will give access to the
  instructors.

## Specification

### Web Architecture

Describe here, with a diagram, the components of your web application and how
they interact. Highlight which parts belong to the application layer, data layer
or presentation layer. How did you ensure that HTML is not rendered server side?

### API

#### REST compliance

Describe here to what extent did you follow REST principles and what are the
reasons for which you might have decided to diverge. Note, you must not describe
the whole API here, just the design decisions.

#### OpenAPI Resource models

Describe here synthetically, which models you have introduced for resources.

### Data model

Describe with an ER diagram the model used in the data layer of your web
application. How these map to the OpenAPI data model?

## Implementation

### Tools used

Describe here which tools, languages and frameworks did you use for the backend
of the application.

### Discussion

Describe here:
```

```
- How did you make sure your web application adheres to the provided OpenAPI
  specification?

- Why do you think your web application adheres to common practices to partition
  the web application (static assets vs. application data)

- Describe synthetically why and how did you manage session state, what are the
  state change triggering actions (e.g., POST to login etc..).

- Which technology did you use (relational or a no-SQL database) for managing
  the data model?

## Other information

### Task assignment

Describe here how development tasks have been subdivided among members of the
group, e.g.:

- Foo worked on front end (80%) and OpenAPI Spec (20% of the time)
- Bar worked on ....

### Analysis of existing API

Describe here if you have found relevant APIs that have inspired the OpenAPI
specification and why (at least two).

### Learning outcome

What was the most important thing all the members have learned while developing
this part of the project, what questions remained unanswered, how you will use
what you've learned in your everyday life?

Examples:

- Foo learned to write SQL queries and Javascript but wanted to know more about
  caching, he's probably going to create his own startup with what she has
  learned
- Bar learned how to deploy on a cloud platform, he would have liked to know
  more about promises for asynchronous code..
```

4.2 Template for emails sent to the instructor concerning problems with the backend

To streamline the resolution of problems, we highly recommend to fill up this form with the description of the issue you are having.

We realize there is a lot of data requested here. We ask only that you do your best to provide as much information as possible so we can better help you.

```
## Team

- Team administrator name; email
- Team member n.1 name; email
- Team member n.2 name; email
```

```
## Relevant links

Put here the link to your repository. Invite one of the teachers as
administrators/members of the team in order for them to be able to
access it.

## What are you trying to achieve

-   Goal: *describe what are you trying to achieve here*

-   Complete, non-ambiguous steps to reproduce the issue from your repo.
    These are the steps that the teachers will follow when trying investigate
    your issue; if it is a command line invocation, please copy and paste the
    complete command with its arguments, options etc..

    1.  *describe step 1 here*

    2.  *describe step 2 here*

    3.  ...

-   Expected results: *what is your expected outcome*

## Describe what you get

-   Actual results: *what outcome do you get instead?*. Remember, "It doesn't
    work" is not a problem statement.

-   Do you get any error message, log, stack trace? put it here

           COPY THE ERROR MESSAGE, INCLUDING STACK TRACE HERE
```

ASSIGNMENT NAME: HYP/Technology/BE
DESCRIPTION: Evaluation of the backend of the application
COURSE: Hypermedia applications
STUDENT/GROUP:
ADDITIONAL CONSTRAINTS: To pass the back-end part, each criteria grade should at least be ”partially” or more. Besides, the web application D0 should be up and running on a remote cloud platform such as Heroku at the time of the exam; no projects with a non-functioning application will be accepted. The back-end documentation D1 should be available as a single page on the same server as D0 and provide links to D2-D5; besides it should fulfill each criteria in the dimensions below.

DOCUMENTATION (30 %)	fully (100%)	partially (60%)	not acceptable (0%)
D1 provides a link to D2, i.e., the YAML file containing the specification of the API	☐	☐	☐
D1 provides a link to D3, i.e., the SwaggerUI page from where tests can be run	☐	☐	☐
D1 provides a link to the online repository D5 (github.com or bitbucket) where application code is stored and a link to a copy of the repo as file D4	☐	☐	☐
D1 provides an overview of the architecture of the application (we suggest a diagram)	☐	☐	☐
D1 provides a short discussion on the OpenAPI resource models and why developers think they have been compliant with REST principles	☐	☐	☐
D1 describes the data model used in the data layer through a conventional (e.g. ER diagram) formalism and the database technology used	☐	☐	☐
D1 contains the list of tools, languages and frameworks used for developing the backend	☐	☐	☐
D1 contains a short discussion on ensuring i) that the web application is adheres to its OpenAPI specification, ii) that it follows common practice on asset partitioning, iii) how session state is managed and what are the state triggering actions and iv) which database technology has been integrated.	☐	☐	☐
D1 describes the tasks assigned to each member of the group	☐	☐	☐
D1 describes and points to relevant real APIs that have inspired the specification	☐	☐	☐
D1 each of the members of the team describes what was the most important thing they’ve learned while developing this part of the project, what questions remained unanswered, how they will use what they’ve learned their everyday life	☐	☐	☐

SPECIFICATION (30 %)	fully (100%)	partially (60%)	not acceptable (0%)
D2 defines REST endpoints that are concise, simple and actions over resources are naturally mapped to HTTP verbs	☐	☐	☐
D2 Defines a concise, simple model for data passed through	☐	☐	☐

IMPLEMENTATION (20 %)	fully (100%)	partially (60%)	not acceptable (0%)
D0 (and D4, D5) adhere to the provided OpenAPI specification	☐	☐	☐
D0 (and D4, D5) Adheres to common practices to partition the web application (static assets vs. application data)	☐	☐	☐
D0 (and D4, D5) Does not render HTML server-side with data coming from the database. This data must be rendered to HTML in the presentation layer.	☐	☐	☐
D0 (and D4, D5) Provides session management enabling authorization and authentication	☐	☐	☐
D0 (and D4, D5) Integrate either a relational or a no-SQL database for managing the data model	☐	☐	☐

TESTING (20 %)	fully (100%)	partially (60%)	not acceptable (0%)
D4 runs as D0 on a remote cloud service (e.g., Heroku) and is usable	☐	☐	☐
D3 contains working executable requests towards the API endpoints (these will be tested during grading so they must work!)	☐	☐	☐