Enunciado

PROGRAMAÇÃO WEB

Trabalho Prático I

Identificador	T1
Designação	Diário de Viagens (Parte I)
Data de lançamento	2017-05-01
Aula livre	2017-05-27
Data de entrega	2017-06-03 às 9h30
Equipa	Grupos constituídos por 3 elementos
Avaliação	A avaliação final do trabalho é expressa através de uma classificação numa escala numérica de 0 a 20 valores.
Critérios de avaliação	30% - Completude da solução apresentada 25% - Respeito pelas boas-práticas de programação de RESTful APIs 15% - Documentação da API usando Swagger 15% - Completude do relatório e qualidade da escrita 15% - Qualidade da apresentação/defesa do trabalho à turma
Entregáveis	1. Código fonte 2. Link para o servidor de testes na Cloud (Heroku) 3. Documentação da API (Swagger) 4. Relatório de 6 páginas segundo o modelo "CS Proceedings da Springer" (PDF) [1] 5. Slides da defesa

DESCRIÇÃO DO TRABALHO

As férias estão aí ao virar da esquina! Neste período, quer se vá relaxar para uma ilha paradisíaca, quer se vá deixar levar pela confusão de uma grande cidade, é natural querer-se guardar as experiências e partilhá-las com os amigos. As redes sociais tradicionais servem esse propósito, mas deixam a desejar no que diz respeito à criação de histórias/diários sobre as viagens e suas experiências, em particular.

Este trabalho tem como objetivo colmatar essa lacuna através da criação de uma rede social focada neste aspeto. Uma aplicação móvel e, posteriormente, uma aplicação Web serão as interfaces de *front-end* para a criação e consulta de diários de viagem, partilha de conteúdo e interação com amigos. Assim, este projeto foca-se essencialmente no desenvolvimento da aplicação de *back-end* que irá suportar as aplicações descritas.

A solução a desenvolver deve implementar uma RESTful API capaz de dar resposta aos requisitos descritos neste documento, permitindo gerir de forma eficaz toda a informação necessária para suportar o funcionamento as aplicações de *front-end*.

[1] Template e instruções disponíveis em ftp://ftp.springer.de/pub/tex/latex/llncs/word/splnproc1703.zip

REQUISITOS FUNCIONAIS

Na primeira fase do trabalho, pretende-se que seja desenvolvida uma API RESTful que permita criar viagens a um dado local do mundo assim como registar experiências vividas em cada um dos dias dessa viagem.

Em concreto, o utilizador deve poder definir uma viagem, identificando a localização de destino (país, cidade, coordenadas GPS, descrição da viagem, etc.) e uma data. Definida a viagem, o utilizador deve poder adicionar momentos/experiências à sua viagem, que ficam guardados na *timeline* dessa viagem.

Um momento/experiência tem sempre uma data e uma designação, uma narrativa (opcional), coordenadas GPS (opcional) e pode ser acompanhada de texto, fotos, vídeos, áudio, condições meteorológicas, bem como, todo o tipo de media que ajude a relembrar o momento ou a experiência.

> *Por exemplo, imaginemos que o utilizador vai a Londres de férias. Experiências típicas são, por exemplo, o embarque no avião, o desembarque, a chegada ao London Marriot Hotel County Hall, o almoço no restaurante Gillray's Steakhouse & Bar onde almoçou um bife grelhado com queijo e batatas fritas, a volta no London Eye, a utilização do metro até à Baker Street Station, a entrada no Madame Tussaud's, ou a foto junto à fomosa porta 221b na Baker Street.*

Nesta fase do trabalho, toda a informação registada será de domínio público, ou seja, não existe necessidade de autenticar utilizadores nem de implementar restrições de acesso.

Para todas as entidades informacionais deverão ser implementadas as 4 operações de CRUD (Create, Read, Update e Delete).

Qualquer utilizador que tenha acesso à API deverá ser capaz de classificar um momento/experiência como:

- Já fui muito feliz aqui!
- Nunca tinha experimentado, estou maravilhado!
- Esta vai direto para minha *bucket list*!

Ao nível da viagem, deverá ser possível obter um somatório das várias classificações associadas a aos momentos/experiências dessa viagem.

A persistência dos dados nesta fase é opcional, ou seja, estes devem ser guardados pelo menos em memória, no entanto, quem desejar preservar o estado da aplicação em bases de dados, ou em ficheiros JSON no disco está livre de o fazer.

REQUISITOS NÃO-FUNCIONAIS

O trabalho deve ser desenvolvido recorrendo às tecnologias lecionadas ao longo da Unidade Curricular, nomeadamente, HTTP (e seus códigos de erro), NodeJS, Express, HTML 5, CSS, etc.

O trabalho deve também seguir uma metodologia assente em planeamento onde se espera que as equipas de trabalho sejam capazes de definir e documentar em sede de relatório técnico os seguintes aspetos inerentes ao processo de desenvolvimento:

1. Entidades informacionais e as suas propriedades
2. Formato dos dados trocados entre clientes e servidor (e.g. JSONP)
3. Esquema de identificadores das entidades informacionais (e.g. UUID)

4. Modelo de persistência de dados (memória, disco, base de dados, etc.)
5. Estruturação da API segundo a metodologia RESTful

Faz parte do espírito da Unidade Curricular de Programação Web e da natureza do próprio curso que os alunos sejam capazes de adquirir novos conhecimentos de forma autónoma. Por este motivo, espera-se que os alunos investiguem e implementem também os seguintes requisitos especiais:

1. Durante a defesa do trabalho, a aplicação deverá encontrar-se num servidor aplicacional na Cloud[2] de modo a evitar problemas do tipo "*It works on my machine!*". Isto facilitará também o desenvolvimento, demonstração e testes da API, bem como a apresentação das aplicações de *front-end*.
2. Os serviços REST deverão ser acompanhados de um conjunto de páginas HTML que documentem a API e que permitam de forma simples testar os vários métodos implementados recorrendo à framework Swagger[3] [4] [5].

O relatório deverá indicar o endereço onde estas interfaces poderão ser consultadas (API REST e documentação em Swagger).

RELATÓRIO

O relatório deverá possuir no máximo 6 páginas segundo o modelo "CS Proceedings da Springer"[6] e conter as seguintes secções:

1. Título, autoria e email
2. Resumo
3. Introdução e objetivos
4. Arquitetura da solução
   4.1 Principais decisões ao nível da estrutura da API, formato dos dados e modelo de persistencia
   4.2 Apontadores para a documentação da API
   4.3 Apontadores para o servidor de testes
5. Conclusões e melhorias futuras
6. Referências

ENTREGÁVEIS

A entrega do trabalho deverá realizar-se através do envio dos materiais identificados no cabeçalho até à data e hora definidas no início deste enunciado. O porta-voz do grupo deverá submeter um ZIP com os vários entregáveis perfeitamente identificados. Não serão aceites trabalhos entregues fora do prazo estabelecido.

---

[2] https://www.heroku.com/nodejs

[3] https://github.com/swagger-api/swagger-node

[4] https://youtu.be/wC5hxY0RItQ

[5] https://youtu.be/xggucT_xl5U

[6] Modelo e instruções disponíveis em ftp://ftp.springer.de/pub/tex/latex/llncs/word/splnproc1110.zip

Durante a defesa do trabalho, todos os membros do grupo deverão estar presentes para defender a solução desenvolvida e aptos para responder a questões sobre o trabalho. Em caso de falta não-justificada[7] aplica-se uma penalização de 50% na nota final do trabalho ao elemento faltoso.

AVALIAÇÃO

Todos os trabalhos serão avaliados com base nos entregáveis e na defesa do trabalho, tendo por base nos critérios de avaliação estabelecidos (ver início do enunciado). A defesa do trabalho realizar-se-á numa sessão plenária com uma duração de 15 minutos acrescidos de 5 minutos para discussão, onde deverá ser feita uma apresentação do trabalho e principais decisões tomadas ao longo do projeto (slides) e uma demonstração ao vivo da solução final (recorrendo ao servidor de testes na Cloud).

A nota final da disciplina atribuída a cada aluno corresponde à seguinte formulação:

Nota final = T1 * 0.3 + T2 * 0.4 + T3 * 0.3
T1 – Trabalho prático 1
T2 – Trabalho prático 2
T3 – Exame ou ensaio (a decidir posteriormente)

A deteção de fraude em algum dos trabalhos implica a reprovação à unidade curricular.

[7] http://www.portaldaempresa.pt/CVE/pt/Geral/faqs/Recursos_Humanos/Faltas/