Desenvolvimento de Aplicações Web - semestre 2020-1
Curso de Engenharia de Computação

Segundo trabalho prático – Persistência de dados cliente e servidor

Enquanto o grupo está definindo o seu projeto e/ou desenvolvendo o protótipo de uma aplicação Web, estamos antecipando o enunciado do segundo *release*. Para a segunda entrega, precisamos fazer evoluir o protótipo do primeiro *release*, considerando a prática de prototipação *corner stone* – em que o código do protótipo é integrado as produto final, contrapondo-se à prática *throw away*, em que o protótipo é descartável. Assim, nesta nova entrega, devem ser incluídos recursos que darão à aplicação características dinâmicas, tanto do lado cliente como do lado servidor da aplicação. Para isso, usaremos o *JavaScript* (JS) nos dois lados da arquitetura. No caso do servidor, a tecnologia a ser usada é o NodeJS (https://nodejs.org/en/) acrescido dos módulos express e ejs.

Além dos aspectos relativos à dinâmica, usaremos também recursos para persistência de dados.

Alguns exemplos com o uso da tecnologia proposta estão disponíveis em *links* no final deste documento. O objetivo é ilustrar as possibilidades de desenvolvimento, auxiliar no aprendizado dos alunos que não conhecem essa tecnologia e fornecer os recursos mínimos para implementação dos requisitos abaixo especificados.

Observações:

- Por mais que seja tentador solucionar e desenvolver o projeto usando ferramentas de produtividade, lembro que a proposta é trabalhar com o mínimo de recursos intermediários que mascarem, de alguma forma, a complexidade das aplicações. O objetivo é que vocês, futuros engenheiros, tenham uma visão mais aprofundada da estrutura e do desenvolvimento de aplicações Web, bem como dos componentes de base desenvolvidos para essa plataforma.
- A sequência dos *releases* propostos certamente não é a mais adequada do ponto de vista da Engenharia de Software. A motivação para a sequência proposta é didática, visto que o desenvolvimento Web requer o conhecimento de muitas tecnologias e precisamos fazer uma opção de como abordá-las.

ENUNCIADO DO 2º RELEASE

1. Como requisitado no primeiro *release*, a aplicação deve considerar mais de um perfil de usuários. Assim, o conteúdo das páginas Web apresentadas para usuários e/ou perfis distintos é necessariamente diferenciado, o que exige dinâmica nos componentes da aplicação. Nesse sentido, desenvolva soluções com as tecnologias sugeridas para:
    a) Carregar as diferentes seções do documento HTML a partir de dados **armazenados no servidor**, considerando usuários e/ou perfis distintos. A sugestão é utilizar arquivos locais (no servidor) com estruturas JSON, mas vocês são livres para usar o MongoDB ou outra tecnologia que utilize estruturas no formato JS.

b) Organize os seus dados de tal maneira que um mesmo ***template*** possa ser usado para diferentes usuários e/ou situações. Os *templates* podem ser definidos usando o EJS, conforme exemplos fornecidos.

c) Implemente pelo menos uma funcionalidade que permita a formatação de páginas a partir de dados **fornecidos explicitamente pelo usuário** por meio de formulário.

d) A carga de conteúdo deve contemplar tanto **requisições HTTP síncronas** como **assíncronas**. Como é impossível especificar *a priori*, neste enunciado, o conteúdo a ser carregado em cada uma dessas modalidades (porque isso depende do projeto a ser apresentado no 1º *release*), o grupo deverá buscar uma forma de implementar as duas abordagens. O objetivo é que todos os participantes do grupo possam conhecer os dois mecanismos.

e) Implemente nesta versão o **armazenamento local (no cliente)** das configurações personalizáveis de *layout*, como os recursos de acessibilidade especificados para o protótipo (tamanho de fonte, cores, etc.). A configuração armazenada para usuários diferentes deverá ser considerada e carregada de maneira automática. Você pode utilizar *cookies* ou objetos *Storage* do JS.

Para dar suporte a execução deste *release*, são fornecidos alguns exemplos de código que podem ser carregados nos links abaixo disponibilizados:

Código (Link para um arquivo .zip)	**Descrição**
Modificação do DOM e carga assíncrona de conteúdo	Código exemplo que permite a alteração da estrutura do documento html carregado no navegador por meio de *scripts* cliente. O exemplo utiliza *JavaScript* no cliente e no servidor. No lado cliente, há dois exemplos para carregar conteúdo por meio de requisições HTTP assíncronas alternativas: um é baseado no uso do objeto *XMLHttpRequest*; outro usa a função *fetch*. No lado servidor, é implementado em *NodeJS* um servidor HTTP simples que lê um arquivo (no próprio servidor) e o carrega no corpo da resposta HTTP. O código está detalhadamente comentado. Outras instruções iniciais podem ser encontradas no comentário do arquivo *loadContent.html*.
Projeto express com diferentes rotas (URLs) e uso de EJS para preprocessamento de scripts	Projeto simples para carga de arquivos fixos segundo um *template* com o uso do EJS. Neste programa, estamos usando um módulo do *NodeJS* que encapsula muitas funcionalidades

Código (Link para um arquivo .zip)	**Descrição**
	necessárias à criação de funções de *back-end* para aplicações Web. Portanto, para executar esse exemplo simples, além do *NodeJS*, é preciso ter instalados os módulos EXPRESS e EJS. O *express* trabalha com o conceito de "rotas", que identificam os recursos a serem requisitados ao servidor pelos clientes. No exemplo, são definidas e detalhadas quatro rotas que interceptam mensagens GET. Três dessas rotas acionam *templates* que irão receber estruturas de dados e formatar, após preprocessamento, a página que será embutida como corpo da resposta HTTP. Para executar o exemplo, é preciso ter o NodeJS no computador e as dependências instaladas. As instruções para essa configuração estão inseridas nos comentários do arquivo *server.js*. Todos os arquivos js e ejs estão comentados detalhadamente.
Projeto com geração dinâmica (no servidor) usando um layout predefinido e uso de cookies	Exemplo simples de geração dinâmica de conteúdos segundo um layout predefinido. Neste exemplo, além do uso do EJS para construção de conteúdo dinamicamente, usa-se um layout comum a um conjunto de páginas. O exemplo é bem rudimentar, mas auxilia à compreensão do uso destes recursos e permite abrir perspectivas para outras criações. Além disso, é exemplificado neste código a geração e uso de *cookies* a partir do cabeçalho *set-cookies* (servidor -> cliente) e cookies (cliente -> servidor) do HTTP. No exemplo, o *cookie* é usado para guardar informações sobre a última visualização do usuário dentre três diferentes "CVs" disponíveis (Fulano, Beltrano e Cicrano).

Data de entrega (depósito no portfólio do Solar do protótipo, eventuais artefatos intermediários) e apresentação (vídeo ou videoconferêcia): 27/08/2020