MCZA019 - Programação para Web
Plano de ensino **ECE**

Prof. Diogo S. Martins
Centro de Matemática, Computação e Cognição
Universidade Federal do ABC

Q1 2020
v.14/04

1 Informações básicas

- TPI: 2-2-4
- Aulas ao vivo:

ter	08-10h	Google Meet ou Zoom	semanal
sex	10-12h	Google Meet ou Zoom	semanal

- Worksite TIDIA-AE: PW-2020Q1
- Atendimento:
- Plantão de dúvidas:

  sex 09-10h Google Meet ou Zoom semanal
- Comunicação com o professor:
  - Prioritariamente no workspace Slack: `pw-2020q1.slack.com`
    Todos foram convidados para o workspace, vide seu email institucional.
  - Por email (caso Tidia indisponível): `santana.martins@ufabc.edu.br`
- Repositório Bitbucket: `https://bitbucket.org/diogo_martins/pw-2020q1/src`

2 Descrição da disciplina

A disciplina aborda os principais aspectos do desenvolvimento de aplicações web full-stack. Trata de temas relevantes ao desenvolvimento de clientes, servidores, documentos estruturados, *scripting*, comunicação síncrona/assíncrona e integração com sistemas de banco de dados. Em profundidade, aplica os conceitos por meio de pilha de tecnologias baseadas em Javascript.

3 Requisitos recomendados

Para participar dessa disciplina é recomendação oficial ter cursado e sido aprovado em:

- BC1501 - Programação Orientada a Objetos
- MC3310 - Banco de Dados

A disciplina também demanda conhecimentos de sistemas distribuídos e engenharia de software, que podem ser compreendidos/reforçados no contexto das aulas.

4 Objetivos

- Familiarizar-se com os fundamentos do desenvolvimento de aplicações Web;
- Ser capaz de projetar arquiteturas de aplicações Web utilizando e refinando padrões de projeto;
- Ser capaz de desenvolver aplicações Web seguindo os princípios de alta coesão e baixo acoplamento.

Ao final do curso, espera-se que o aluno, aprovado com conceito satisfatório, possua habilidades que permitam, a partir de um conjunto de requisitos, projetar aplicações Web que sejam eficientes, confiáveis, seguras e de fácil manutenção.

5 Bibliografia

1. Sebesta, Robert W. *Programming the World Wide Web*, 8th edition. Pearson Addison Wesley, 2014.
2. Connolly, R. e Hoar, R. *Fundamentals of Web Development*, 1st edition. Pearson, 2014.
3. Deitel, Paul J., Deitel, Harvey M. e Deitel, A. *Internet & World Wide Web: How to Program*, 5th edition. Prentice Hall, 2011.
4. Haverbeke, M. *Eloquent Javascript: A Modern Introduction to Programming*, 3rd. edition. No Starch Press, 2018.
5. Freeman, E. e Robson, E. *Head First Javascript Programming: A Brain-Friendly Guide*, 1st edition. O'Reilly Media, 2014.
6. Materiais online (tutorias, guias, referências, etc.) trazidos no contexto de cada aula.

6 Metodologia de ensino-aprendizagem (com adaptações ECE)

Com a introdução do ECE, aponta-se as seguintes alterações na metodologia:

- As aulas serão síncronas, ao vivo, duas vezes por semana, em horário coincidente com as aulas presenciais (vide seção 1);
- As aulas serão gravadas e as gravações disponibilizadas após cada aula;
- O plantão de dúvidas será assíncrono via Slack;
- As atividades formativas tornam-se somativas, isto é, agora as atividades precisam ser entregues e valem nota.

A metodologia fundamental dessa disciplina é a aprendizagem ativa[1], segundo a qual a responsabilidade pelo aprendizado é do aprendiz. Essa metodologia enfatiza que as atividades em aula devem estimular a análise, a síntese e a avaliação de conteúdos. Difere em grande medida da aprendizagem passiva, na qual as atividades em aula geralmente demandam apenas o consumo passivo de conteúdos de palestras, vídeos ou leituras. Os métodos de ensino-aprendizagem que empregraremos são a *aula invertida*[2] e a *aprendizagem baseada em problemas*[3].

O tempo de aula será usado essencialmente para a resolução de problemas relacionados ao conteúdo abordado. Na maior parte dos casos, os problemas envolverão a construção de uma aplicação com base em um conjunto de requisitos. Naturalmente, não podemos ignorar os aspectos teóricos do conteúdo, portanto utilizaremos o método da aula invertida — cada tema constitui um ciclo, e cada ciclo é formado pelas seguintes atividades:

1. **Estudo prévio ou posterior**: alunos estudam os conteúdos do ciclo, com base nos recursos indicados pelo professor, antes ou depois da aula, a depender do tema;
2. **Prática**: resolução de problemas ou mini-projetos sobre o conteúdo estudado, durante ou após a aula ao vivo;

[1] Para uma introdução breve sobre essa metodologia, vide `https://en.wikipedia.org/wiki/Active_learning`
[2] `https://en.wikipedia.org/wiki/Flipped_classroom`
[3] `https://en.wikipedia.org/wiki/Problem-based_learning`

3. **Avaliação somativa**: após a aula, é necessário entregar uma atividade desenvolvida para a verificação de conhecimento ou propor uma resolução para um problema relacionado;

Sobre as aulas ao vivo: nas aulas ao vivo executaremos tutoriais no tema da aula, em formato similar ao que já fazíamos nas aulas presenciais. Idealmente, recomenda-se que os alunos tentem implementar os exemplos junto com o professor, para ficar mais claro eventuais dificuldades e dúvidas durante a aula.

7 Criterios de avaliação somativa

A avaliação somativa consiste nos componentes dados pela Equação 1, onde:

$$N_F = 0.3 \cdot N_{atv} + 0.4 \cdot N_{prova} + 0.3 \cdot N_{proj} \tag{1}$$

- N_{atv} é a nota de 75% das somativas com as melhores notas. Como total para calcular a porcentagem, consideramos todas as somativas enunciadas durante o ECE (provavelmente 14);
- N_{prova} é a nota da Prova;
- N_{proj} é a nota do projeto final.

As provas são individuais e consistem na construção de uma aplicação, completa ou parcialmente, com base em um conjunto de requisitos enunciados. A depender da complexidade do tema, é possível que uma prova seja organizada em fases, parte extra-classe e parte em classe — essa dinâmica será propriamente enunciada na ocasião mais adequada para a prova.

O conceito final será obtido de acordo com a Equação 2.

$$C_F = \begin{cases} \texttt{A}, \text{ se } N_F \in [8.5, 10.0] \\ \texttt{B}, \text{ se } N_F \in [7.0, 8.5) \\ \texttt{C}, \text{ se } N_F \in [5.5, 7.0) \\ \texttt{D}, \text{ se } N_F \in [5.0, 5.5) \\ \texttt{F}, \text{ se } N_F \in [0.0, 5.0) \\ \texttt{O}, \text{ se ausência exceder } 25\% \end{cases} \tag{2}$$

Sobre as atividades

Não teremos mais avaliações formativas. Todas as avaliações no ECE serão *somativas*.

Substituição das formativas por atividades somativas: é possível entregar as formativas anteriores, convertendo-as em somativas. Como apenas parte das somativas será considerada para nota (vide seção 7), essa estratégia funcionará como avaliação substitutiva desse componente.

Caberá ao aluno demonstrar disciplina e organização de tempo para executá-las, visto que são planejadas para preencher o componente I (4 horas/sem.) previsto no nosso ementário.

Sobre a prova

A prova será presencial, em data a ser marcada posteriormente, pois ainda não sabemos quando as aulas presenciais retornarão. Será divulgada com antecedência em data dentro do calendário de atividades presenciais previsto no ECE.

Sobre o projeto

As entregas do projeto permanecem no mesmo formato, com as datas ajustadas ao novo calendário. A apresentação do projeto será online, no formato de reunião virtual.

8 Critérios de avaliação para atividades de programação (inclusive as somativas)

Tanto as provas, quanto o projeto e as atividades somativas, qualificam-se como atividades de programação. No caso das atividades que envolvam prazo de entrega, não serão aceitas as que forem entregues fora do prazo. A nota máxima de cada atividade será obtida apenas se a mesma for entregue no prazo e executada correta e completamente.

A nota do projeto final será calculada de acordo com um conjunto de critérios, que encontra-se publicado em documento específico de enunciado do projeto. Adicionalmente, o projeto deverá ser entregue incrementalmente, de acordo com calendário a ser divulgado naquele mesmo documento, sendo que cada entrega corresponde a uma fração da nota total do projeto.

Os programas solicitados em atividades de avaliação serão analisados quanto aos seguintes critérios gerais:

- **Eficiência**: os programas desenvolvidos deverão ter bom desempenho, o que pode englobar o tratamento adequado dos seguintes fatores:
  - ler e escrever dados nas quantidades mínimas necessárias para resolver o problema;
  - não desperdiçar memória primária (RAM), tanto no cliente como no servidor;
  - acessar memória secundária (disco, banco de dados, etc.) somente quando necessário e sem redundância;
  - minimizar comunicação em rede, tanto em tempo de operação quanto em quantidade de dados transmitidos (i.e. não baixar os mesmos dados múltiplas vezes, fazer o máximo de trabalho possível/viável no cliente, etc.);
  - entre outros.
- **Acurácia**: o programa deverá atender adequadamente a todos os requisitos enunciados para a atividade;
- **Estrutura e organização do código**: atentar principalmente aos seguintes aspectos:
  - **Auto-documentação**: nomes intuitivos para variáveis e métodos/funções;
  - **Modularização**: funções/métodos com alta concisão e baixo acoplamento, isto é, que sejam em sua maioria curtos, e que realizem preferencialmente uma única tarefa;
  - **Comentários**: documentação completa porém ao mesmo tempo concisa (sem poluição visual, apenas nos lugares adequados e necessários). Programar é redigir numa linguagem formal de alto nível, que será interpretada tanto por computadores quanto por humanos.

8.1 Mecanismos de avaliação substitutivos

Teremos dois mecanismos de avaliação substitutiva:

- Para as atividades, como consideraremos 75% das maiores notas, para substituir basta entregar com atraso até atingir no mínimo 75% de entregas;
- As atividades pré-ECE (antigas formativas, números 01-07), podem ser entregues até o último dia de aula, sem desconto de nota. As somativas posteriores terão prazo, portanto entregas com atraso implicarão em desconto de nota;
- Para a prova, marcaremos prova substitutiva presencial no período de atividades presenciais do cronograma ECE (ainda não foi definido);

8.2 Mecanismo de recuperação

A recuperação será aplicada apenas aos alunos que tiverem conceito final D ou F. Ocorrerá em data a ser divulgada posteriormente, no período de atividades presenciais do cronograma ECE.

A nota obtida na prova de recuperação (N_R) será usada para obter a nota final com recuperação (N_{FR}), que consiste na média estabelecida pela Equação 3.

$$N_{FR} = \frac{N_F + N_R}{2} \tag{3}$$

O conceito final obtido na recuperação (C_{FR}) é o conceito que entrará no histórico, obtido de acordo com os limiares para a nota final de recuperação (N_{FR}) dados pela Equação 4.

$$C_{FR} = \begin{cases} \texttt{C}, \text{ se } N_{FR} \geq 5.5 \\ \texttt{D}, \text{ se } N_{FR} \in (5.0, 5.5) \\ \texttt{F}, \text{ se } N_{FR} \leq 5.0 \end{cases} \tag{4}$$

No caso dos alunos que não participarem da recuperação, $C_{FR} = C_F$.

9 Cronograma de aulas

O plano a seguir pode variar de acordo com o aproveitamento aferido nas turmas durante o quadrimestre.

Cronograma pré-ECE

As datas marcadas com (+) são feriados.

Aula #	Dia	Tema
1	11/02	Arquitetura da Web
2	14/02	HTML I
3	18/02	CSS I
4	21/02	HTML II
-	25/02+	Feriado (Carnaval)
5	28/02	CSS II + Bootstrap
6	03/03	Javascript
7	06/03	Javascript
8	10/03	JQuery
9	13/03	Ajax

Cronograma ECE

- Em todas as datas marcadas, ocorrerão aulas ao vivo (lives) no horário da aula;
- Nas datas marcadas com asterisco, por serem feriados, é possível que não haja aula ao vivo, porém nessas situações uma aula gravada será disponibilizada para estudo assíncrono.

Aula #	Dia	Tema
10	21/04*	Apresentação ECE e revisão
11	24/04	Estudos de caso
12	28/04	Node.js
13	01/05*	Node.js
14	05/05	Express
15	08/05	Express
16	12/05	MongoDB
17	15/05	MongoDB

Continua na próxima página...

Aula #	Dia	Tema
18	19/05	Sessões
19	22/05	Sessões
20	26/05	SQL
21	29/05	SQL
22	02/06	Acompanhamento dos projetos
23	05/06	Apresentação dos projetos
-	-	Prova: semana 01 do período de reposição
-	-	Sub: semana 02 do período de reposição
-	-	Rec: semana 03 do período de reposição

10 Código de honra

A aprovação na disciplina é baseada exclusivamente no esforço e trabalho pessoal do discente, ao qual cabe garantir que não ajudará ou receberá ajuda não-permitida em qualquer atividade usada pela equipe docente para fins de avaliação (e.g. provas, trabalhos, listas, etc.).

Exemplos de violação do código de honra incluem:

- Copiar atividades avaliativas (e.g. listas, trabalhos, provas, etc.) ou permitir que outros discentes copiem suas atividades avaliativas;
- Colaboração não-permitida entre indivíduos ou grupos (e.g. oferecer vantagens em troca de soluções prontas, doar trechos para o trabalho de outro grupo, etc.);
- Permitir que outros assumam sua identidade em atividades avaliativas (e.g. entregar trabalho que não fez ou permitir que outros façam provas por você);
- Plagiarismo (i.e. aplicável a textos, programas de computador, etc.);
- Receber ou conceder ajuda em atividades avaliativas quando o contexto mostra que não é sensato receber tal ajuda.

Como consequências de violação do código de honra tem-se:

- Reprovação automática na disciplina, com conceito F;
- Denúncia na Comissão de Transgressões Disciplinares Discentes da Graduação, a qual decidirá sobre a punição adequada à violação, o que pode levar a advertência, suspensão ou desligamento, de acordo com os arts. 78-82 do Regimento Geral da UFABC.