MCZA019 - Programação para Web
Plano de ensino

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

Q2 2023
v. 29/05

1 Informações básicas

- TPI: 2-2-4
- Horários oficiais:

seg	21-23h	S-214-0	semanal
qui	19-21h	409-2	semanal

- Comunicação online com o professor:
  - Mensagens no Moodle
  - Email: santana.martins@ufabc.edu.br
- Plantão de dúvidas (presencial):

  seg 18-19h S528-2
- Sala no Moodle: pw-2023-q2

  https://moodle.ufabc.edu.br/course/view.php?id=1228

  Todos já foram convidados para a sala, em caso de dificuldade de acesso, entrar em contato com o professor.

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, programação assíncrona e integração com sistemas de banco de dados. Em profundidade, aplica os conceitos por meio de pilha de tecnologias baseadas em Javascript/Typescript.

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. Connolly, R. e Hoar, R. Fundamentals of Web Development, 1st edition. Pearson, 2014.
2. Deitel, Paul J., Deitel, Harvey M. e Deitel, A. Internet & World Wide Web: How to Program, 5th edition. Prentice Hall, 2011.
3. Sebesta, Robert W. Programming the World Wide Web, 8th edition. Pearson Addison Wesley, 2014.
4. Materiais online (tutoriais, guias, referências, etc.) trazidos no contexto de cada aula.

6 Metodologia de ensino-aprendizagem

Indissociabilidade entre teoria e prática. As aulas combinam os conteúdos de teoria e prática com dinâmica fluída, ou seja, teremos breves resumos teóricos seguidos de sessões mais longas de programação com introdução de novos conceitos teóricos à medida que mostrem-se necessários nos contextos mais adequados.

Aprendizagem ativa. [1] A dinâmica da aula enfatiza a análise, a síntese e a avaliação dos conteúdos, em detrimento do consumo passivo de conhecimento pronto. A abordagem segue a linha do "aprender fazendo" ao invés do "aprender olhando". Por esta razão, reforça-se a importância de, sempre que possível, seguir os tutoriais de programação sincronamente às aulas. Além disso, a dinâmica das aulas depende da iniciativa individual do aluno, o qual deve tornar-se protagonista do seu aprendizado, resolvendo os problemas e trazendo suas dúvidas, caso contrário seu aproveitamento nas atividades presenciais será baixo.

Dinâmica das atividades presenciais. Dispomos de um dia de teoria (em sala de aula) e um dia de prática (em laboratório). Dada essa configuração, utilizaremos a infra-estrutura dos espaços físicos do seguinte modo:

- Aulas teóricas. Como não dispomos de computadores individuais, teremos, além da discussão dos conteúdos teóricos, também exercícios focados na análise de códigos prontos, sessões de programação, bem como a discussão de dúvidas levadas pelos alunos.
- Aulas práticas. Como neste espaço há disponibilidade de computadores individuais, teremos exercícios regulares de programação, com enunciados disponíveis no Moodle. As atividades serão sempre formativas, i.e., não valerão nota. Porém, a entrega das atividades no Moodle contará como frequência. O tempo em aula pode ser usado também para a discussão de dúvidas.

[1] https://en.wikipedia.org/wiki/Active_learning

7 Avaliação

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

$$N_F = 0.4 \cdot N_{atv} + 0.6 \cdot N_{proj} \tag{1}$$

- N_{atv} é a média das atividades somativas;
- N_{proj} é a nota do projeto.

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

$$C_F = \begin{cases} \text{A, se } N_F \in [8.5, 10.0] \\ \text{B, se } N_F \in [7.0, 8.5) \\ \text{C, se } N_F \in [6.0, 7.0) \\ \text{D, se } N_F \in [5.0, 6.0) \\ \text{F, se } N_F \in [0.0, 5.0) \\ \text{O, se ausência exceder 25\%} \end{cases} \tag{2}$$

A frequência na disciplina será mensurada via:

- Frequência de entrega das avaliações (atividades e prova).
- Frequência de acesso aos conteúdos no Moodle.

Sobre as atividades

Ideia geral As atividades avaliativas são de execução individual e consistem na construção de aplicativos, completa ou parcialmente, com base em um conjunto de requisitos enunciados. Teremos um total de 4 atividades somativas, ou seja, valendo nota. As atividades servem de problemas-base motivadores para estudar o material indicado bem como praticar as técnicas num programa concreto. A carga horária das atividades é de 24h por aluno (50% do componente "I").

Auto-avaliação formativa. As atividades contarão, sempre que possível, com recursos de verificação automática de corretude e estilo de programação, com o objetivo de auxiliar o estudante a detectar, de modo automatizado, problemas recorrentes de programação. Na ausência de recursos de verificação automática, o estudante contará com um checklist para analisar a aderência do seu resultado ao que é esperado. Cabe ao aluno garantir que o programa passe por todas as verificações, sejam manuais ou automáticas. Além disso, nas atividades em que haja checklist, deve garantir que todos os critérios sejam atendidos. Submissões com falhas terão descontos substanciais na nota.

Avaliação somativa. Além dos testes automáticos, o professor analisará as submissões em atenção aos aspectos qualitativos e éticos detalhados neste plano de ensino. Ou seja, passar em todos os testes automáticos não implica necessariamente em nota máxima, visto que análise posterior pode implicar em desconto parcial ou total na nota da atividade.

Prazos. Cada atividade tem prazo de ao menos 15 dias para entrega. Não serão aceitas atividades entregues com atraso.

Nro.	Tema	Abertura	Fechamento
1	HTML e CSS	01/06	15/06
2	Javascript	15/06	29/06
3	Programação Assíncrona	29/06	13/07
4	Single Page Application	13/07	27/07
5	Server Side Rendering	27/07	17/08

Sobre o projeto

A dinâmica do projeto está especificada em documento separado ("Instruções para o projeto final"), o qual encontra-se disponível no Moodle. A carga horária do projeto é de 24h por aluno (50% do componente I).

Sobre o sistema de submissão

Tanto as atividades quanto o projeto serão submetidos via Moodle e/ou via GitHub Classroom. O conteúdo da segunda semana de aula envolve treinamento sobre o uso combinado destas duas plataformas de submissão.

8 Critérios de avaliação de programas

A nota máxima de cada atividade será obtida apenas se a mesma: i) for entregue no prazo; ii) estiver completa (fez todos os itens solicitados); iii) estiver correta (cada item está correto).

A nota do projeto final será calculada de acordo com um conjunto de critérios próprios, que encontra-se publicado em documento específico de enunciado do projeto.

Os programas solicitados em atividades de avaliação serão submetidos aos seguintes tipos de verificações:

- Verificações automáticas. Testes de corretude, de estilo de programação, de erros comuns e de detecção de plagiarismo. Falhas em quaisquer subconjuntos dos testes implicam em descontos de nota, exceto para os testes de plagiarismo, cujas falhas implicam nas sanções previstas na seção "Código de honra".
- Verificações manuais. O professor inspecionará os programas para verificar os 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;
    - Do lado do cliente, minimizar comunicação em rede, tanto em tempo de operação quanto em quantidade de dados transmitidos (i.e. não baixar os mesmos arquivos dados múltiplas vezes — exceto nos casos em que o browser faça caching, fazer o máximo de trabalho possível/viável no cliente ou no servidor antes de transmitir, etc.);
  - Acurácia: o programa deverá atender adequadamente a todos os requisitos enunciados para a atividade;
  - Corretude: o programa deverá atender a todos os requisitos enunciados e a solução deverá estar funcionalmente correta;
  - Estrutura e organização do código: atentar principalmente aos seguintes aspectos:
    - Auto-documentação: nomes intuitivos para variáveis e métodos/funções;
    - Modularização: funções/métodos/classes com alta coesão 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).
  - Autenticidade: o código é original e não foi copiado de outras fontes (i.e. web, livros, terceiros, etc.), sob pena das sanções previstas no código de honra.

8.1 Mecanismos de avaliação substitutivos

Não se aplica, pois não temos provas. No caso de atestados que durem por muitos dias, prejudicando as entregas na disciplina, a flexibilização dos prazos será caso-a-caso, via negociação com o professor.

8.2 Mecanismo de recuperação

A prova de recuperação será aplicada apenas aos alunos que tiverem conceito final D ou F. A prova será remota, via Moodle. Seguirá a metodologia de avaliação baseada em projetos. Consiste na construção de uma aplicação, completa ou parcialmente, com base em um conjunto de requisitos enunciados. A prova será organizada em duas fases:

- Na parte 1 (semana 1 do Q2.2023), o estudante terá no máximo 1 semana para familirizar-se com o código-base do sistema, e possivelmente implementar algumas partes faltantes;
- Na parte 2 (semana 2 do Q2.2023), com duração de 2 horas, na semana agendada para a prova (vide cronograma), em dia e horário a escolha do aluno, serão implementadas funcionalidades adicionais, com base no código da parte 1.

No final, será feita apenas uma entrega, dentro do prazo previsto no Moodle, correspondente aos resultados das partes 1 e 2. O prazo será controlado automaticamente pelo sistema de submissão. Não serão aceitas provas atrasadas ou entregues por meios de submissão alternativos.

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.

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

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

9 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.

10 Cronograma de aulas

O plano a seguir pode variar de acordo com o aproveitamento aferido na turma durante o quadrimestre.

Aula #	T/P	Data	Tema
1	29/05	T	Introdução ao desenvolvimento Web
2	01/06	P	HTML
3	05/06	T	CSS
	08/06		Feriado (reposição em 21/08)
4	12/06	T	Design responsivo
5	15/06	P	Javascript
6	19/06	T	Javascript
7	22/06	P	Typescript
8	26/06	T	Typescript
9	29/06	P	Programação assíncrona I
10	03/07	T	Programação assíncrona II
11	06/07	P	Padrões de projeto para Web
12	10/07	T	Single Page Applications (SPA)
13	13/07	P	Single Page Applications (SPA)
14	17/07	T	Backend as a Service (BaaS)
15	20/07	P	Desenvolvimento server-side e Node.js
16	24/07	T	Bancos de dados e MongoDB
17	27/07	P	Camada Model com MongoDB
18	31/07	T	Servidores com Node e Express
19	03/08	P	Web services REST
20	07/08	T	Server-side rendering (SSR)
21	10/08	P	Server-side rendering (SSR)
22	14/08	T	Sessões Web
23	17/08	P	Sessões Web e autenticação
24	21/08	T	Prazo final para entrega do projeto