MCZA019 - Programação para Web
Plano de ensino

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

QS 2020
v.21/09

1 Informações básicas

- TPI: 2-2-4
- Horários oficiais:

ter	21-23h	semanal
qui	19-21h	semanal

- Plantão de dúvidas:

  qui 18-19h Sala Discord

- Comunicação com o professor:
  - Prioritariamente na sala Discord. Link de convite: `https://discord.gg/pz3jwDG`
    Usar o link para se inscrever. Preferencialmente, usar seu nome real no perfil, para facilitar a comunicação.
    Os plantões são nossos únicos eventos síncronos. A comunicação será via voz, texto, vídeo e compartilhamento de tela, com o objetivo de esclarecer dúvidas e/ou reforçar temas vistos nas videoaulas e outros materiais.
  - Por email (caso Discord indisponível): `santana.martins@ufabc.edu.br`

  As comunicações com o professor, exceto no horário de plantão, serão assíncronas, em geral com resposta dentro de 24h em dias úteis.

- Sala no Moodle: `pw-2020qs`
  `https://moodle.ufabc.edu.br/course/view.php?id=808`
  Todos foram convidados para a sala, verifique seu email institucional.

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/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

Formato das aulas. As aulas serão assíncronas, em formato de vídeo-aula, duas aulas por semana. Em semana de prova, reduziremos a frequência para apenas uma aula. Cada aula será disponibilizada no respectivo dia e horário oficial da turma. O conteúdo englobará tutoriais, os quais, idealmente, devem ser seguidos de forma síncrona, ou seja, que os alunos tentem implementar os exemplos ao mesmo tempo que em assistem ao vídeo. Essa abordagem tende a tornar mais explícitas as eventuais dificuldades e dúvidas.

Interações. Em duas modalidades: i) assíncrona, via Discord canal de texto, com prazo de 24 horas para resposta (em dias úteis); e ii) síncrona, via Discord canal de voz, no horário de plantão de dúvidas (vide seção 1).

Técnicas de ensino aprendizagem:

- **Aprendizagem ativa**[1]. As atividades das videoaulas enfatizam a análise, a síntese e a avaliação dos conteúdos, ou seja, haverá menor ênfase no consumo passivo de conteúdos;
- **Aprendizagem baseada em problemas**[2]. O conteúdo das aulas é estruturado fundamentalmente em problemas-base concretos, os quais servem de contexto para abordar os temas conceituais/abstratos da ementa;

[1] https://en.wikipedia.org/wiki/Active_learning
[2] https://en.wikipedia.org/wiki/Problem-based_learning

- **Aula invertida**[3]. O tempo da aula será utilizado essencialmente para a resolução dos problemas-base. A aquisição do conhecimento teórico-conceitual é abordado via estudo prévio ou posterior com os materiais (referências de estudo) indicados no roteiro da respectiva aula. Como as referências de estudo são divulgadas juntamente com as videoaulas, cabe ao aluno decidir se deve estudá-las antes ou depois de assitir aos vídeos. Embora cada aula inclua uma sucinta revisão/contextualização dos aspectos teórico-conceituais, é de fundamental importância que haja o estudo individual das referências de estudo para garantir a profundidade do tema.

Ciclos. O curso é estruturado em ciclos, sendo que cada ciclo dura uma semana e é composto dos seguintes momentos:

1. **Estudo prévio ou posterior**. Alunos estudam os conteúdos do ciclo, com base nos recursos indicados pelo professor (textos, vídeos, tutoriais interativos, etc.), antes ou depois da aula, a depender do tema;
2. **Aula**. Resolução de problemas ou mini-projetos sobre o conteúdo estudado, durante a aula online;
3. **Avaliação formativa**. As atividades para entrega contarão, quando possível, com recursos para verificação automática de corretude e boas práticas de programação, ou mesmo rubricas para auto-avaliação manual, as quais devem ser analisadas e garantidas pelos alunos antes da submissão.
4. **Avaliação somativa**. Após a aula, é necessário entregar uma atividade desenvolvida para a verificação de conhecimento ou propor uma resolução para um problema relacionado. Adicionalmente, teremos duas avaliações globais: prova individual e trabalho em grupo.

7 Avaliação

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

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

- N_{atv} é a média de 75% das atividades somativas com as melhores notas. Como total para calcular a porcentagem, consideramos todas as atividades enunciadas durante o quadrimestre (por volta de 12);
- N_p é a nota da prova;
- N_{proj} é a nota do projeto.

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

Teremos por volta de uma atividade por semana, com prazo de entrega de 7 dias. Todas as atividades são somativas, ou seja, valem nota. As atividades serão entregues no ambiente Github Classroom. O conteúdo do ciclo 1 engloba treinamento sobre o uso da plataforma. 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 oficialmente para a disciplina.

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 uma rubrica

[3] `https://en.wikipedia.org/wiki/Flipped_classroom`

para analisar a aderência do seu resultado ao que é esperado. Cabe ao aluno garantir que o programa passe por *todas* as verificações automáticas. Além disso, nas atividades em que haja rubrica, deve garantir que todos os critérios sejam atendidos. Submissões com falhas terão descontos substanciais na nota. Em ambos os casos, convencionamos como critérios de auto-avaliação:

- **Satisfatório**: solucionou todos os problemas/requisitos da atividade;
- **Pouco satisfatório**: solucionou alguns problemas/requisitos da atividade;
- **Insatisfatório**: não tentou fazer a atividade.

Sobre a prova

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**, com início na semana 10, será construída a primeira parte da aplicação, com funcionalidades básicas;
- Na **parte 2**, com duração de 3 horas, durante a semana 12, em dia e horário a escolha do aluno, serão implementadas funcionalidades adicionais, com base na implementação obtida na parte 1.

No final, será feita apenas uma entrega, na data prevista no cronograma, 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 dinâmica específica da prova será divulgada junto aos respectivos enunciados no momento adequado.

Sobre o projeto

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.

Critérios de avaliação

Todas as avaliações somativas, isto é, atividades, prova e projeto, qualificam-se como atividades de programação. A nota máxima de cada avaliação será obtida apenas se a mesma for entregue no prazo e executada correta e completamente.

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. Parte dos testes (por volta de 70%) serão distribuídos juntamente com as atividades, de modo que os alunos podem executá-los localmente antes de entregar. A outra parte dos testes (por volta de 30%) serão privados, i.e., não serão divulgados para os alunos.
- **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) somente quando necessário e sem redundância;
    * 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;
  - **Corretude**: o programa deverá passar em todos os testes publicados pelo professor, bem como deverá apresentar boa cobertura (por volta de 80%) de testes nos módulos adicionais implementados pelo aluno;
  - **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 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).
  - **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.

7.1 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;
- No caso da prova, que é assíncrona com janela de uma semana, é desnecessário o mecanismo de substituição, visto que o aluno pode escolher o melhor dia/horário para efetuar a prova.

7.2 Mecanismo de recuperação

A recuperação será aplicada apenas aos alunos que tiverem conceito final D ou F. Ocorrerá na data estabelecida no cronograma, em formato similar ao estabelecido para as outras provas.

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, $C_{FR} = C_F$.

8 Cronograma de aulas

Cada semana corresponde a um ciclo. O cronograma a seguir pode variar de acordo com o aproveitamento aferido nas turmas durante o quadrimestre.

Semana #	Tema
1	Arquitetura da Web + HTML
2	HTML + CSS
3	Typescript
4	DOM

Continua na próxima página...

Semana #	Tema
5	Ajax
6	React
7	Node.js
8	Express
9	JSON + Redis
10	MongoDB + Sessões
11	SQL
12	Revisão + Prova
13	Prova de recuperação

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.), o que envolve copiar porções significativas de textos ou programas de terceiros, sem atribuição de autoria;
- 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.