MCZA033 - Programação Avançada para Dispositivos Móveis
Plano de ensino

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

Q1 2023
v. 06/02

1 Informações básicas

- TPI: 0-4-4
- Modalidade: presencial
- Aulas:

ter	19-21h	409-2	semanal
sex	21-23h	409-2	semanal

- Ferramentas de comunicação:
  - Discord
    Para cadastrar-se na sala, utilizar link disponível no Moodle.
  - Comunicação com o professor
    - Prioritariamente via Discord, por mensagem pública ou privada
    - Por email (caso Discord esteja indisponível): `santana.martins@ufabc.edu.br`
- Plantões de dúvidas:
  - Interações síncronas:

ter	18-19h
sex	18-19h

    Aonde:
    - Se remoto, no Google Meet, com agendamento prévio no Google Calendar da disciplina (vide link de agendamento no Moodle).
    - Se presencial, na sala 528-2.
  - Interações assíncronas:
    Aonde: sala Discord
    Horário: livre, com resposta dentro de 24h em dias úteis.
- Sala no Moodle: `padm-2023.1-na`
  `https://moodle.ufabc.edu.br/course/view.php?id=4248`
  Todos já foram convidados para a sala, em caso de dificuldade de acesso, entrar em contato com o professor.

2 Descrição da disciplina

O desenvolvimento de aplicações para dispositivos móveis (e.g. smartphones, tablets, dispositivos vestíveis, etc.) demanda o tratamento de requisitos que são inerentes à computação móvel, entre os quais pode-se citar: *i)* tamanho reduzido de tela; *ii)* meios de interação limitados para entrada e saída de dados; *iii)* funcionamento restrito à duração de bateria; *iv)* conectividade potencialmente intermitente com o deslocamento geográfico; *v)* poder de processamento inferior a outros dispositivos de computação pessoal contemporâneos (e.g. desktops, notebooks, etc.); *vi)* utilização em múltiplos contextos no cotidiano de seus usuários; entre outros. Com vistas a essas particularidades, a disciplina trata dos principais aspectos do desenvolvimento para dispositivos móveis. Em abrangência, aborda o desenvolvimento de aplicações com atenção aos requisitos específicos de computação móvel, ao mesmo tempo em que reforça conceitos de projeto de software aplicáveis a outras classes de computação. Em profundidade, promove a verificação e experimentação desses conceitos com base na plataforma Android.

3 Requisitos recomendados

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

- MCTA001 - Algoritmos e Estruturas de Dados I
- MCTA018 - Programação Orientada a Objetos
- MCZA019 - Programação para Web

4 Objetivos

- Familiarizar-se com os fundamentos do desenvolvimento de aplicações para dispositivos móveis;
- Ser capaz de projetar aplicações para dispositivos móveis utilizando e refinando padrões de projeto;
- Ser capaz de desenvolver aplicações que usem adequadamente os recursos dos dispositivos móveis.

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 para dispositivos móveis que sejam eficientes, confiáveis, seguras e de fácil manutenção.

5 Ementa

- Conceitos fundamentais de programação móvel;
- Projeto de interfaces para aplicações móveis;
- Armazenamento de dados para aplicações móveis;
- Integração com recursos dos dispositivos;
- Integração com outros sistemas: comunicação cliente-servidor;
- Desenvolvimento para dispositivos heterogêneos (smartphones vs. tablets).

6 Bibliografia

- Literatura da plataforma Android. URL: (`http://developer.android.com/develop/index.html`). Acesso: Jan. 2023.
- Para as obras complementares, consultar o registro da disciplina no Projeto Pedagógico do Bacharelado em Ciência da Computação.

7 Metodologias de ensino-aprendizagem

Aula invertida[1]**.** O tempo da aula será utilizado essencialmente para a resolução de problemas e discussão de dúvidas. A aquisição dos conhecimentos teórico-conceitual e prático é abordado via estudo prévio com os materiais indicados no roteiro da respectiva aula, os quais serão compostos, basicamente, de vídeo-aulas e textos. Cada aula é estruturada em ciclos, cada ciclo composto dos seguintes momentos:

1. **Estudo prévio**. Alunos estudam os conteúdos da aula, com base nos recursos indicados pelo professor (textos, vídeos, tutoriais interativos, etc.), antes da aula. Os recursos serão disponibilizados com antecedência para que haja tempo hábil para o estudo prévio;
2. **Problemas e dúvidas**. Durante as aulas práticas, teremos solução de problemas ou mini-projetos sobre o conteúdo estudado e discussão de dúvidas relacionadas ao material de estudo.

Material de estudo prévio. Principal conteúdo teórico-conceitual do curso, de consumo obrigatório, compostos por vídeo-aulas e textos. O conteúdo engloba, além de aulas com exposição teórica, também tutoriais com práticas de programação, os quais, idealmente, devem ser seguidos de forma síncrona, ou seja, que os alunos tentem implementar os exemplos ao mesmo tempo em que assistem ao vídeo.

Aprendizagem ativa[2]**.** As atividades das videoaulas enfatizam 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 seguir os tutoriais de programação sincronamente aos vídeos. 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 tende a ser baixo.

8 Avaliação

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

$$N_F = 0.5 \times N_{ativ} + 0.5 \times N_{proj} \tag{1}$$

- N_{ativ} é a média das atividades avaliativas;
- N_{proj} é a nota do projeto final.

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 Avaliativas

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. É possível entregar atividades atrasadas, sendo que a data-limite para entrega das atividades consta no cronograma. Atividades atrasadas terão desconto de nota proporcional ao atraso.

Datas-limite para entrega de atividades *sem* descontos:

- Atividade 1: 05/03
- Atividade 2: 26/03
- Atividade 3: 16/04
- Atividade 4: 09/05

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.

9 Critérios de avaliação de programas

A nota máxima de cada atividade de programação 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);
    * Acessar memória secundária (memória flash interna ou externa) 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.);
  - **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, métodos, classes, interfaces e pacotes;
    - **Modularização**: métodos e 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.

9.1 Mecanismos de avaliação substitutiva

Teremos dois mecanismos de avaliação substitutiva:

- Para as atividades, como consideramos 75% das maiores notas, para substituir, basta entregar com atraso até atingir no mínimo 75% de entregas;
- Tanto para as atividades quanto para o projeto, os descontos por atraso podem ser abonados desde que se apresente justificativa formal (vide ConsEPE 227).

9.2 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.

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

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

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**.

11 Cronograma de aulas

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

Aula #	Dia	Tema
01	07/02	Computação móvel
02	10/02	Plataforma Android
03	14/02	Aplicativos
04	17/02	Kotlin
	21/02	Feriado (reposição 04/05)
05	24/02	Kotlin
06	28/02	Interface do usuário
07	03/03	Interface do usuário
08	07/03	Interface do usuário
09	10/03	Interface do usuário
10	14/03	Interface do usuário
11	17/03	Arquiteturas
12	21/03	Arquiteturas
13	24/03	Arquiteturas
14	28/03	Estudo de caso
15	31/03	Estudo de caso
16	04/04	Clientes Web
	07/04	Feriado (reposição 08/05)
17	11/04	Clientes Web
18	14/04	Clientes Web

Continua na próxima página...

Aula #	Dia	Tema
19	18/04	Armazenamento local
	21/04	Armazenamento local
20	25/04	Plataformas como serviço
21	28/04	Plataformas como serviço
22	04/05	Plataformas como serviço
23	08/05	Finalização e entrega do projeto
24	09/05	Finalização e entrega de atividades atrasadas