UNIVERSIDADE FEDERAL DE ALFENAS

DAVID VINÍCIUS ALVES

AKADEMI: APLICAÇÃO DISTRIBUÍDA PARA AUXILIAR ALUNOS E PROFESSORES NA GESTÃO ACADÊMICA

Alfenas/MG
2020

DAVID VINÍCIUS ALVES

AKADEMI: APLICAÇÃO DISTRIBUÍDA PARA AUXILIAR ALUNOS E PROFESSORES NA GESTÃO ACADÊMICA

Trabalho de conclusão de curso apresentado ao curso de Ciência da Computação, da Universidade Federal de Alfenas como requisito parcial à obtenção do título de Bacharelado em Ciência da Computação.

Orientador: Rodrigo Martins Pagliares.

Alfenas/MG
2020

DAVID VINÍCIUS ALVES

AKADEMI: APLICAÇÃO DISTRIBUÍDA PARA AUXILIAR ALUNOS E PROFESSORES NA GESTÃO ACADÊMICA

A Banca examinadora abaixo-assinada, aprova o Trabalho apresentado como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação pela Universidade Federal de Alfenas.

_____

Prof. Rodrigo Martins Pagliares

Universidade Federal de Alfenas

_____

Profa. Mariane Moreira de Souza

Universidade Federal de Alfenas

_____

Prof. Ricardo Menezes Salgado

Universidade Federal de Alfenas

Alfenas/MG
2020

AGRADECIMENTOS

A Deus por ter permitido e abençoado grandemente todo meu caminho até esse momento na minha vida.

Aos meus pais Vitor e Célia que me incentivaram e deram todo o apoio necessário para meu desenvolvimento pessoal e profissional.

Aos meus irmãos Maicon e Erika que sempre estiveram do meu lado em todos os momentos.

À minha namorada Thaís que me ajudou e auxiliou durante todo o período da minha formação.

Ao meu cunhado Rodrigo e meu amigo Ademilson que me auxiliaram diversas vezes durante o desenvolvimento deste trabalho.

RESUMO

A popularização do uso de dispositivos móveis pelos diversos setores da sociedade contribui para que as pessoas ganhem tempo nas tarefas do cotidiano. Em particular, universitários e professores têm, a cada dia, menos tempo disponível para tarefas burocráticas de suas vidas acadêmicas, como gestão de calendário, notas e provas. Neste contexto, aplicativos para gestão da vida acadêmica podem atuar como facilitadores, ajudando professores e universitários a acessarem informações de interesse de uma maneira mais rápida e simples. Embora existam algumas soluções no mercado para gestão acadêmica, poucas possuem versões para dispositivos móveis dos seus sistemas. Assim sendo, alunos e professores têm dificuldade em acessar seu sistema acadêmico, quando possível, pelo celular. Este trabalho tem como objetivo desenvolver uma ferramenta de *software* para gestão acadêmica em universidades. Demos o nome de Akademi para a ferramenta. A principal contribuição do Akademi para alunos e professores é agilizar tarefas diárias relacionadas à gestão da vida acadêmica. Como resultado deste trabalho, obtivemos um produto funcional que visa auxiliar alunos e professores na gestão acadêmica, melhorando a organização e disponibilidade de conteúdo.

Palavras-chave: Gestão Acadêmica. Dispositivos móveis. *API Web*. Aplicativos. Desenvolvimento de software.

ABSTRACT

The popularization of the use of mobile devices by the several sectors of society contributes to people gaining time in daily tasks. In particular, university students and teachers have less time available each day for bureaucratic tasks in their academic lives, such as calendar management, grades and tests. In this context, mobile applications for academic life management can act as facilitators, helping teachers and university students to access information of interest in a faster and simpler way. Although there are some solutions on the market for academic management, few of them have mobile versions of their systems. Therefore, students and teachers have difficulty accessing their academic system, when possible, by cell phone. This work aims to develop a software tool for academic management in universities. We named our tool Akademi. Akademi's main contribution to students and teachers is to streamline tasks related to academic life management. As a result of this work, we obtained a functional product that aims to assist students and teachers in academic management, improving the organization and availability of content.

Keywords: Academic Management. Mobile devices. Web API. Mobile Applications. Software development.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

HTTP	Hypertext Transfer Protocol
API	Application Programming Interface
UML	Unified Modeling Language
RESTful	Representational State Transfer
SSMS	SQL Server Management Studio
SGBD	Sistema Gerenciador de Banco de Dados
RF	Requisitos Funcionais
RNF	Requisitos Não Funcionais
IIS	Internet Information Services
JSON	JavaScript Object Notation
ID	Identificador
SQL	Structured Query Language

SUMÁRIO

1. INTRODUÇÃO

1.1 Motivação

Com a popularização dos dispositivos móveis é de se esperar um crescimento do número de aplicativos desenvolvidos no intuito de facilitar o cotidiano das pessoas (SILVA; SANTOS, 2014).

Considerando que os aplicativos para dispositivos móveis estão se destacando no cotidiano, ante a sua praticidade e eficiência, empresas têm investido no desenvolvimento de *software* para celulares que satisfaçam seus clientes.

Apesar do crescimento do número de aplicativos, poucos são aqueles voltados para gestão acadêmica.

Desta forma, a principal vantagem de se criar um sistema de gestão acadêmica para ser utilizado por meio de celulares é que permite aos alunos e professores terem, a qualquer momento, acesso ao seu sistema acadêmico.

1.2 Problema

Um problema que pode ser enfrentado por alunos e professores é ter que acessar as informações acadêmicas no sistema da instituição via navegador *web* usando celular, uma vez que nem sempre estão com computador em mãos. Além disso, muitos websites não se adaptam bem quando exibidos em aparelho móvel (leiaute não responsivo), pois podem perder a qualidade na distribuição do conteúdo e/ou trazer uma certa lentidão na navegação, como, por exemplo, o Terminal Informativo Unifenas (TIU WEB, 2020) e o Sistema Acadêmico da Unifal-MG (SISTEMA ACADÊMICO UNIFAL, 2020). Isto se dá pelo fato de que os *websites* não possuem acesso aos componentes de *hardware* e *software* próprios do dispositivo da mesma forma que os aplicativos feitos para esse fim possuem, ou não foram projetados para serem acessados por meio de celulares.

De maneira resumida, acessar sites da universidade com leiautes não responsivos através do navegador do celular, bem como a escassez de aplicativos criados para o meio acadêmico, são problemas enfrentados nesta monografia.

1.3 Objetivo

Este trabalho teve como objetivo desenvolver uma ferramenta de *software*, nomeada Akademi, para uso em instituições de ensino superior. Akademi é uma ferramenta distribuída, constituída de um módulo servidor na *web* e de um módulo cliente implantado em dispositivos móveis (aplicativo).

1.4 Resultados

Como resultado deste trabalho, nós criamos um *software* que conta com um módulo aplicativo disponibilizado para dispositivos móveis Android (PEREIRA; DA SILVA, 2009) e um serviço web criado usando a tecnologia ASP.NET (Microsoft, 2020a), responsável por lidar com a busca e gravação de informações em um banco de dados SQL Server (Microsoft, 2020b).

1.5 Metodologia

Para a construção do Akademi, estudamos algumas ferramentas utilizadas atualmente na Universidade Federal de Alfenas: o Sistema Acadêmico, Moodle (MOODLE, 2020) e o Google Classroom (GOOGLE, 2020), além de outras ferramentas de *software* disponíveis na Internet.

A partir deste estudo identificamos as funcionalidades que são comuns entre as ferramentas. Tais funcionalidades foram utilizadas como ponto de partida para o desenvolvimento do Akademi.

1.6 Organização desta monografia

Esta monografia está organizada da seguinte maneira: o Capítulo 2 apresenta as tecnologias usadas na implementação do Akademi. O Capítulo 3 faz uma análise dos trabalhos relacionados com o Akademi. O Capítulo 4 apresenta os atores e objetivos, requisitos funcionais e não funcionais, a arquitetura do Akademi, e alguns diagramas UML que facilitam o entendimento do projeto. O Capítulo 5 apresenta alguns trechos de código presentes no aplicativo e na aplicação *web*. O Capítulo 6

realiza a conclusão deste trabalho assim como os trabalhos futuros para melhoria do Akademi. No Capítulo 7 apresentamos as referências.

2. TECNOLOGIAS USADAS

Este trabalho utiliza diversas tecnologias para desenvolvimento de *software*. Estas tecnologias são descritas neste capítulo.

De maneira resumida, as tecnologias podem ser classificadas em tecnologias para o desenvolvimento de aplicativos para dispositivos móveis e tecnologias para o desenvolvimento de aplicações *web*.

2.1. Tecnologias para desenvolvimento de aplicativos para dispositivos móveis

Akademi conta com um módulo aplicativo desenvolvido para os sistemas operacionais Android e IOS (GOADRICH, 2011), mas atualmente apenas disponível para os sistemas Android, pois não consideramos viável um investimento, por agora, na licença de desenvolvedor da Apple para realização de testes e publicação do aplicativo na sua loja. Usamos a estrutura de interfaces chamada Xamarin.Forms (Microsoft, 2020c), incorporada ao ambiente de desenvolvimento integrado Visual Studio (Microsoft, 2020d), por meio da linguagem de programação C# (TRIGO, 2018). Optamos por estas tecnologias devido à portabilidade oferecida, permitindo escrever aplicativos que podem ser instalados e executados tanto nas plataformas Android quanto IOS, apesar de, no momento, não estar disponível para IOS.

2.2. Tecnologias para desenvolvimento de aplicações *web*

Para lidar com as solicitações do aplicativo Akademi, utilizamos ASP.NET Core, uma estrutura de *software* livre, escrita em C# e multi-plataforma, que permite a criação de serviços RESTful (*Representational State Transfer* - Transferência de Estado Representacional). Os serviços RESTful, também conhecidos como APIs Web (*Application Programming Interface* - Interface de Programação de Aplicações), usam classes ASP.NET Core conhecidas como controladores que derivam de ControllerBase (Microsoft, 2020e).

Para o armazenamento dos dados inseridos no aplicativo, nós utilizamos o Sistema Gerenciador de Banco de Dados Relacionais (SGBD) Microsoft SQL Server, gerenciado pela ferramenta SSMS (SQL Server Management Studio), um ambiente integrado para gerenciar qualquer infraestrutura de SQL (Microsoft, 2020f). O SSMS

fornece ferramentas que auxiliam na configuração e gerenciamento da instância do SQL Server e bancos de dados do módulo servidor deste projeto.

3. TRABALHOS RELACIONADOS

Este capítulo apresenta alguns trabalhos relacionados que, de alguma forma, contribuem com a realização deste trabalho.

O “Minha UFLA” (UFLA, 2017) é um *software* usado na Universidade Federal de Lavras com o propósito de auxiliar alunos e professores por meio da oferta de diversas funcionalidades como, visualizar notas, visualizar saldo e cardápio do restaurante universitário.

O Moodle é uma das soluções em educação mais conhecidas e presente nas instituições. Trata-se de um *software* multi-plataforma gratuito de ensino e aprendizagem. O Moodle conta com um módulo web e outro módulo aplicativo que disponibiliza seus cursos e treinamentos mesmo quando não conectado na Internet do celular.

O Escola em Movimento (ESCOLA EM MOVIMENTO, 2020) oferece um aplicativo escolar que tem como foco a comunicação da instituição com colaboradores, pais e alunos. O aplicativo é utilizado em instituições de ensino do berçário ao curso superior.

O Sistema Acadêmico UNIFAL-MG, é uma aplicação *web* desenvolvida para substituir o antigo sistema da universidade que se tornou inviável por suas limitações, por exemplo, impossibilidade de alunos consultarem suas notas e frequências de aulas.

O Acadêmico Online (SOLUTECH, 2020) é um *software* voltado para instituições de ensino, que consiste em uma ferramenta multi-plataforma integrada com os diversos setores de uma universidade como, biblioteca, secretaria, financeiro, aplicativo móvel e etc.

O Google Classroom é uma sala de aula *online* do Google. Trata-se de uma ferramenta gratuita que pode ser acessada tanto por computadores como celulares. Por meio dessa ferramenta alunos e professores podem se comunicar e manter as aulas à distância mais organizadas.

O Escola Web (ESCOLA WEB, 2020) é uma solução para instituições de ensino que atendem seus alunos tanto presencial quanto à distância. Suas funcionalidades abrangem diversos departamentos das instituições além do acadêmico, como o financeiro e a biblioteca.

O Quadro 3.1 apresenta a relação entre as funcionalidades dos aplicativos apresentados neste capítulo com as do Akademi. Observamos que a proposta do Akademi incorpora a maioria das funcionalidades previstas em outros aplicativos. E outra informação é que as funcionalidades “Visualizar notas” e “Visualizar disciplinas” são comuns entre todas as ferramentas analisadas.

Para comparação apresentada no Quadro 3.1, desconsideramos as funcionalidades que não se relacionam com o departamento acadêmico da universidade, por exemplo, baixar boletos para pagamento de parcelas, que é referente ao departamento financeiro da universidade.

Quadro 3.1 - Funcionalidades levantadas de trabalhos relacionados ao Akademi.

Funcionalidades	Akademi	Escola em Movimento	Moodle	Minha UFLA	Sistema Acadêmico UNIFAL	Acadêmico Online	Google Classroom	Escola Web
Visualizar disciplinas	X	X	X	X	X	X	X	X
Visualizar frequência do aluno	X	X			X	X		X
Enviar e receber mensagens de *chat*	X	X	X					X
Realizar atendimento ao aluno					X	X		X
Anexar arquivos		X	X			X	X	X
Visualizar horário de aulas	X			X	X	X	X	X
Gerenciar crédito restaurante				X				
Visualizar perfil de usuários	X	X	X				X	X
Visualizar itinerário de transporte público	X			X				
Visualizar notas	X	X	X	X	X	X	X	X
Visualizar data de entrega de atividades	X		X				X	X
Realizar chamada	X		X		X			X

Fonte: Elaborado pelo autor.

4. AKADEMI

Este capítulo descreve as características do *software* Akademi, tais como atores e objetivos; requisitos funcionais (RFs) e não funcionais (RNFs); arquitetura utilizada; e diagramas UML exemplificativos.

4.1. Introdução

Akademi é uma aplicação *web* com componentes distribuídos em dois módulos: um módulo (aplicativo, atuando como *frontend*) e um módulo servidor (aplicação *web)*.

O módulo aplicativo, implantado em dispositivos móveis, fornece experiência de uso com o usuário.

O módulo servidor, implantado em um servidor de aplicação fornece a lógica de negócios e persistência de dados.

Desenvolvemos o Akademi objetivando fornecer uma solução/ferramenta para instituições de ensino com finalidade de melhorar e facilitar o acesso de alunos e professores às informações do dia a dia acadêmico, por meio de uma interface criada especificamente para celulares e utilizando tecnologias recentes para desenvolvimento de aplicativos.

4.2. Principais atores e objetivos

O *software* Akademi é composto de dois atores, *Aluno* e *Professor*. O Quadro 4.1 demonstra de modo geral os objetivos dos atores.

Quadro 4.1 - Principais objetivos dos atores do *software* Akademi.

Código	Ator	Objetivos
OBJ-01	Aluno/Professor	Acessar o aplicativo por meio de *email* e senha
OBJ-02	Aluno/Professor	Visualizar informações das disciplinas
OBJ-03	Aluno	Consultar notas e frequência
OBJ-04	Aluno	Consultar Itinerário de transporte público que passa por sua instituição
OBJ-05	Professor	Gerenciar informações nas disciplinas que ministra, tais como notas, atividades, etc.,
OBJ-06	Professor	Gerenciar tarefas semanais para organização de sua agenda

Fonte: Elaborado pelo autor.

4.3. Requisitos funcionais

De acordo com Sommerville (2011, pág. 59), as declarações de serviço a respeito de como o sistema deve reagir a entradas e a determinadas situações e, ainda, explicitar o que não se deve fazer, é chamada de requisitos funcionais.

Requisitos Funcionais (RFs) são de extrema importância para construção de *software*, pois caracterizam as necessidades ou exigências do *software*, normalmente identificadas do ponto de vista de um usuário.

Parte da lista de RFs implementada no Akademi pode ser vista no Quadro 4.2. O Anexo A apresenta uma lista detalhada com todos RFs implementados atualmente pela ferramenta. O quadro está distribuído da seguinte maneira: a primeira coluna identifica o código do requisito funcional, a coluna “Objetivo” está relacionada com o código dos objetivos dos atores presentes no Quadro 4.1; "Requisito" descreve o RF e, por fim, a coluna descrição fornece maiores detalhes acerca do requisito.

Os RFs apresentados no Quadro 4.2 são utilizados em outras seções deste trabalho, como direcionamento para modelos de análise, *design* e implementação.

Quadro 4.2 - Requisitos funcionais usados como direcionamento para análise, *design* e implementação do Akademi.

Código	Objetivo	Requisito	Descrição
RF14	OBJ-05	Realizar chamadas	Deve ser possível que professor realize chamada nas aulas.
RF09	OBJ-02	Visualizar notas	Deve ser possível que o aluno veja suas notas nas disciplinas. Assim como o professor verá as notas de seus alunos em cada atividade das disciplinas que ministra.

Fonte: Elaborado pelo autor.

4.4. Requisitos não funcionais

Os Requisitos Não Funcionais (RNFs), como definido por Sommerville (2011, pág.59), são restrições aos serviços ou funções oferecidos pelo sistema, como tempo de resposta, restrições no processo de desenvolvimento e aquelas impostas pelas normas que, muitas vezes, aplicam-se como um todo no sistema, diferentemente das características individuais ou serviços do sistema.

Os RNFs estão diretamente associados à qualidade do sistema entregue ao usuário, tais como confiabilidade, tempo de resposta e tecnologia de persistência de dados (SOMMERVILLE, 2011). Para melhor entendimento, o Quadro 4.3 apresenta os principais RNFs do Akademi.

Quadro 4.3 - Requisitos não funcionais do Akademi.

Categoria	Código	Descrição
Desempenho	RNF01	As informações dos alunos e professores não devem demorar mais que 30 segundos para serem exibidas na tela.
Disponibilidade	RNF02	O dispositivo deve estar conectado a um serviço de Internet para utilizar as funções do aplicativo.
Usabilidade	RNF06	Acesso às principais informações do usuário deve acontecer priorizando a menor quantidade de cliques.
Interoperabilidade	RNF07	O acesso de todas as informações dos usuários acontece, via formato JSON, pela integração com uma API.

Quadro 4.3 - Requisitos não funcionais do Akademi.

Segurança	RNF03	A autenticação do usuário no sistema deve ser baseada em *token* que expira em 12 horas, após esse tempo é necessário acessar novamente o sistema através de um *login*.
	RNF04	A senha do usuário deve ser criptografada.
	RNF05	Os requisições feitas ao servidor são aceitas mediante autorização do perfil do usuário.
Compatibilidade	RNF08	O Akademi aplicativo é compatível com dispositivos que possuam o sistema operacional Android e IOS.
Internacionalização	RNF09	Inicialmente o Akademi possui apenas a versão em português.

Fonte: Elaborado pelo autor.

4.5. Arquitetura

A arquitetura do Akademi é composta basicamente por um módulo aplicativo e um módulo servidor. O módulo servidor contém uma aplicação web desenvolvida em camadas, com banco de dados e lógica de negócios expostos via HTTP (*Hypertext Transfer Protocol*) por meio de uma API web.

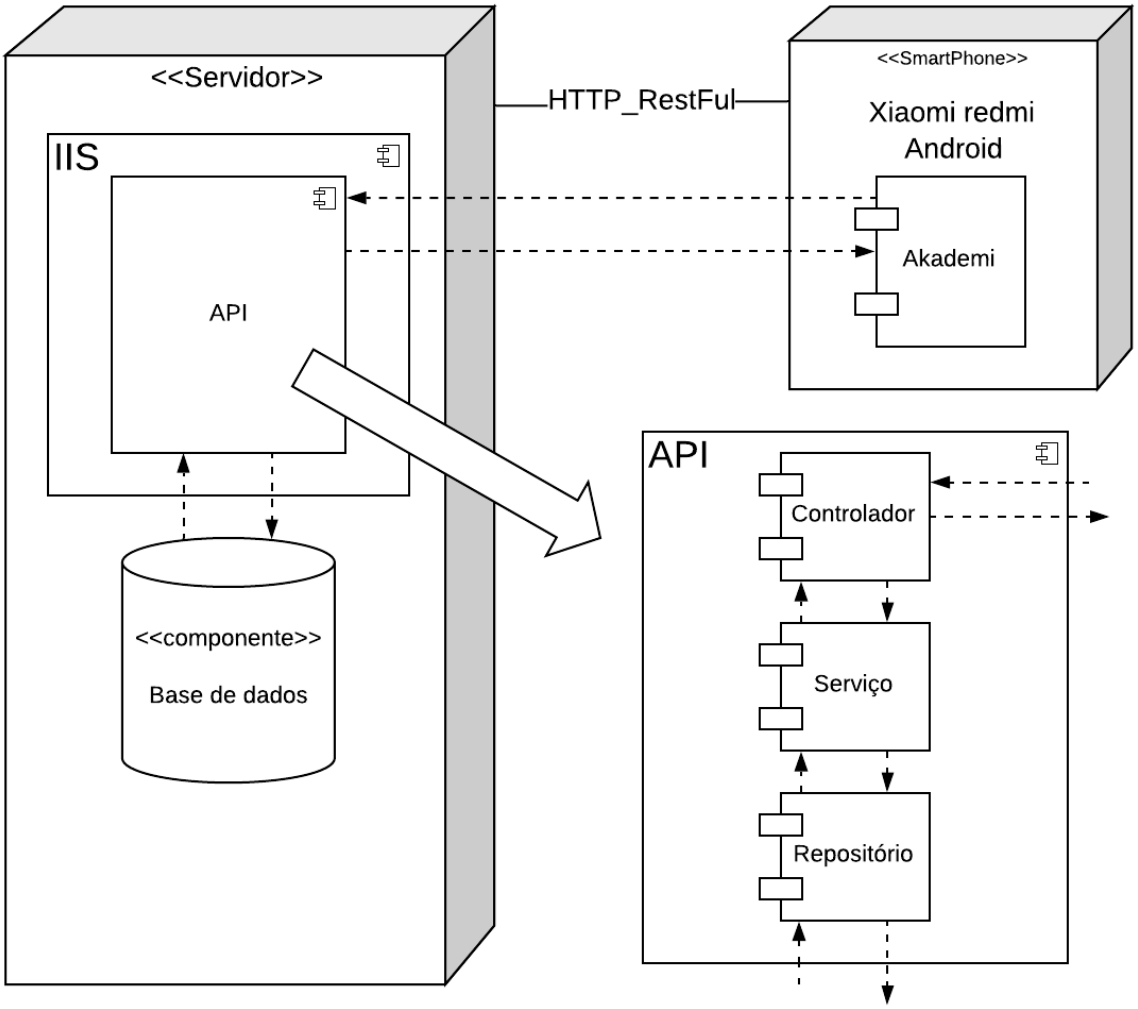
O aplicativo é instalado em um dispositivo móvel com o sistema operacional Android com no mínimo a versão API *level* 21. Cabe ao aplicativo a responsabilidade de realizar todas as chamadas ao módulo servidor.

O servidor hospeda o banco de dados, que contém todos os dados referentes a alunos e professores que utilizam o Akademi, e a API web, responsável pela integração entre aplicativo, a lógica de negócios do Akademi e o banco de dados. É importante ressaltar que há um servidor *web* IIS (Internet Information Services) que engloba todas as camadas da API, sendo responsável por aceitar todos as requisições HTTP feitos pelo aplicativo e integrar as tecnologias necessárias utilizadas para criação da API *web*.

Conforme pode ser visto na Figura 4.1, após o aluno acessar o aplicativo em busca de suas notas, por exemplo, uma requisição HTTP GET é enviada para a API *web* e aceita pelo IIS. A primeira camada da API *web* que recebe tal requisição é o *controlador* (vide parte inferior direita na Figura 4.1). Compete ao controlador a implementação dos métodos associados aos verbos HTTP (GET, POST, PUT,

DELETE, etc), delegando controle para a camada serviço e direcionando a busca na camada repositório. A camada repositório acessa o banco de dados e obtêm as notas do aluno, retornando as mesmas para a camada serviço ordenada por data de atividade. A camada de serviço retorna os dados para a camada *controlador* que serializa a lista para o formato JSON (*JavaScript Object Notation*) antes de enviar como resposta para o aplicativo que, ao receber, exibe as notas na tela para o aluno.

Figura 4.1 - Diagrama de implantação UML (UML *deployment diagram*) da arquitetura do Akademi.



Fonte: Elaborado pelo autor.

4.6. Diagrama de classes em nível de projeto (*design*) e Diagramas de Sequência UML

Além do diagrama de Implantação UML discutido na seção anterior, este trabalho utiliza os diagramas UML de Classe e Sequência. O diagrama de classes documenta a estrutura estática do Akademi. Os diagramas de Sequência tem como objetivo documentar o comportamento interno do Akademi a partir de trocas de mensagens dos objetos que colaboram entre si para realização de RF (consultar nota, realizar chamada, entre outros).

4.6.1 Diagrama de Classes UML em nível de projeto

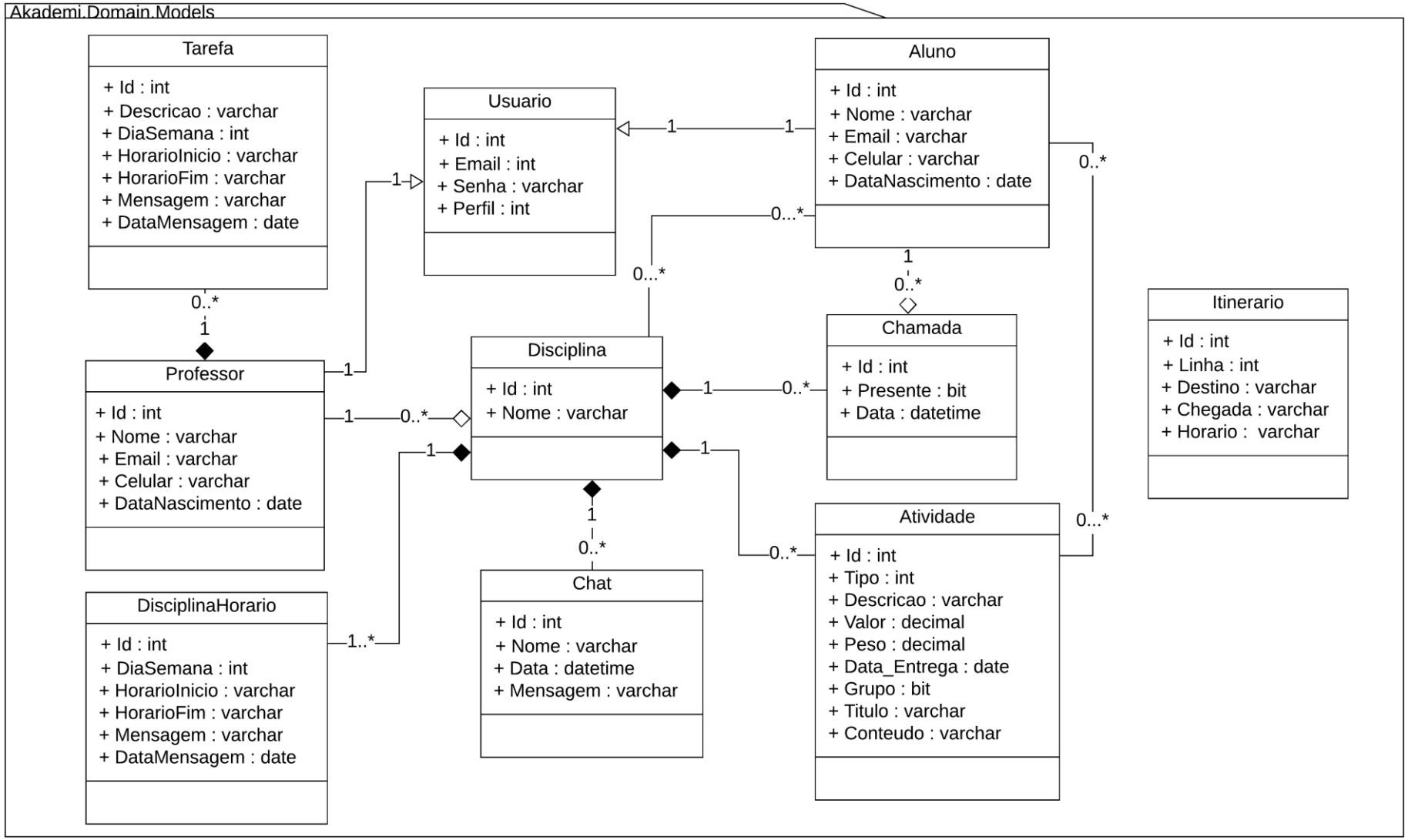
A Figura 4.2 apresenta o diagrama de classes em nível de projeto (LARMAN, 2007) criado para o Akademi. Quando usamos um diagrama de classes em nível de projeto e não em um nível conceitual, representamos conceitos específicos de tecnologias de implementação, como por exemplo, atributos *id* para representar um identificador único do objeto quando persistido em banco de dados por meio de um *framework* objeto relacional (usamos Microsoft Entity Framework Core (Microsoft, 2020g)).

Conforme pode ser visto na figura 4.2, o diagrama de classes do Akademi possui duas relações de generalização entre a classe Usuário com as classes Professor e Aluno.

Outro relacionamento presente no diagrama é o de composição, que existe entre Disciplina e as classes Chamada, Atividade, Chat e DisciplinaHorario, na composição as classes relacionadas à Disciplina só podem ser criadas após existir uma Disciplina e deixam de existir se a Disciplina for excluída.

A título de esclarecimento, para melhor leitura do diagrama de classes, omitimos os métodos *get* e *set*.

Figura 4.2 - Diagrama de classes UML do Akademi.



Fonte: Elaborado pelo autor.

4.6.2 Diagrama de Sequência UML

Para uma visão mais detalhada do comportamento interno ao Akademi, a partir da colaboração entre objetos de software, utilizamos o Diagrama de Sequência UML. Os RFs do Akademi que são abordadas pelos Diagramas de Sequência (Figuras 4.3 e 4.4) são Realizar Chamada e Consultar Notas, ambas descritas no Quadro 4.2.

Para melhor entendimento e simplicidade partimos do pressuposto que o usuário (Aluno ou Professor) já tenha realizado o acesso ao sistema por meio de *e-mail* e senha. Também omitimos a apresentação gráfica do retorno das chamadas para melhor leitura do diagrama.

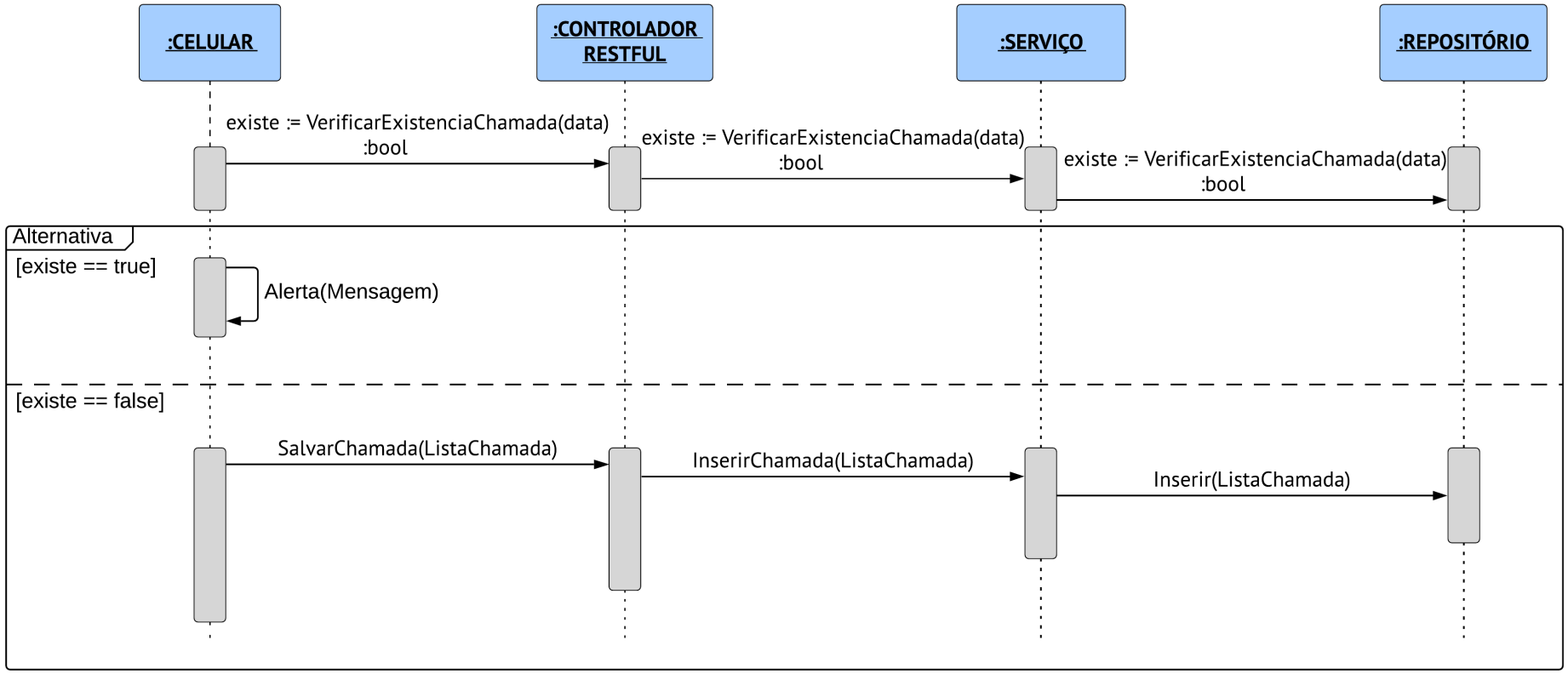
A Figura 4.3 exibe o Diagrama de Sequência para o RF14 (Realizar Chamadas), em que podemos observar que para realizar uma chamada primeiramente o Akademi faz uma verificação no banco por meio do método `VerificarExistenciaChamada()` para saber se já existe uma chamada com a data que o *Professor* está tentando cadastrar. Sendo assim, se já existir uma chamada, o retorno é verdadeiro e é apenas exibida uma mensagem de alerta (parte superior do quadro - Alternativa), mas caso não exista chamada, o retorno da verificação é falso e o passo seguinte é a chamada do método `SalvarChamada()` que tem como parâmetro a `ListaChamada`, informada pelo *Professor* após acessar a tela de realização de chamada, que contém as informações dos alunos presentes e faltantes utilizadas para o cadastro de uma chamada (parte inferir do quadro - Alternativa).

Em sequência, a Figura 4.4 exibe o Diagrama de Sequência para o RF09 (Visualizar Notas). Nesse segundo diagrama, o processo é mais simples, tendo apenas um comportamento diferenciado para os dois perfis de usuários presentes no Akademi, *Aluno* e *Professor*.

Para o *Aluno*, o retorno do método `BuscarNotas()` é uma lista com todas atividades da disciplina e suas respectivas notas.

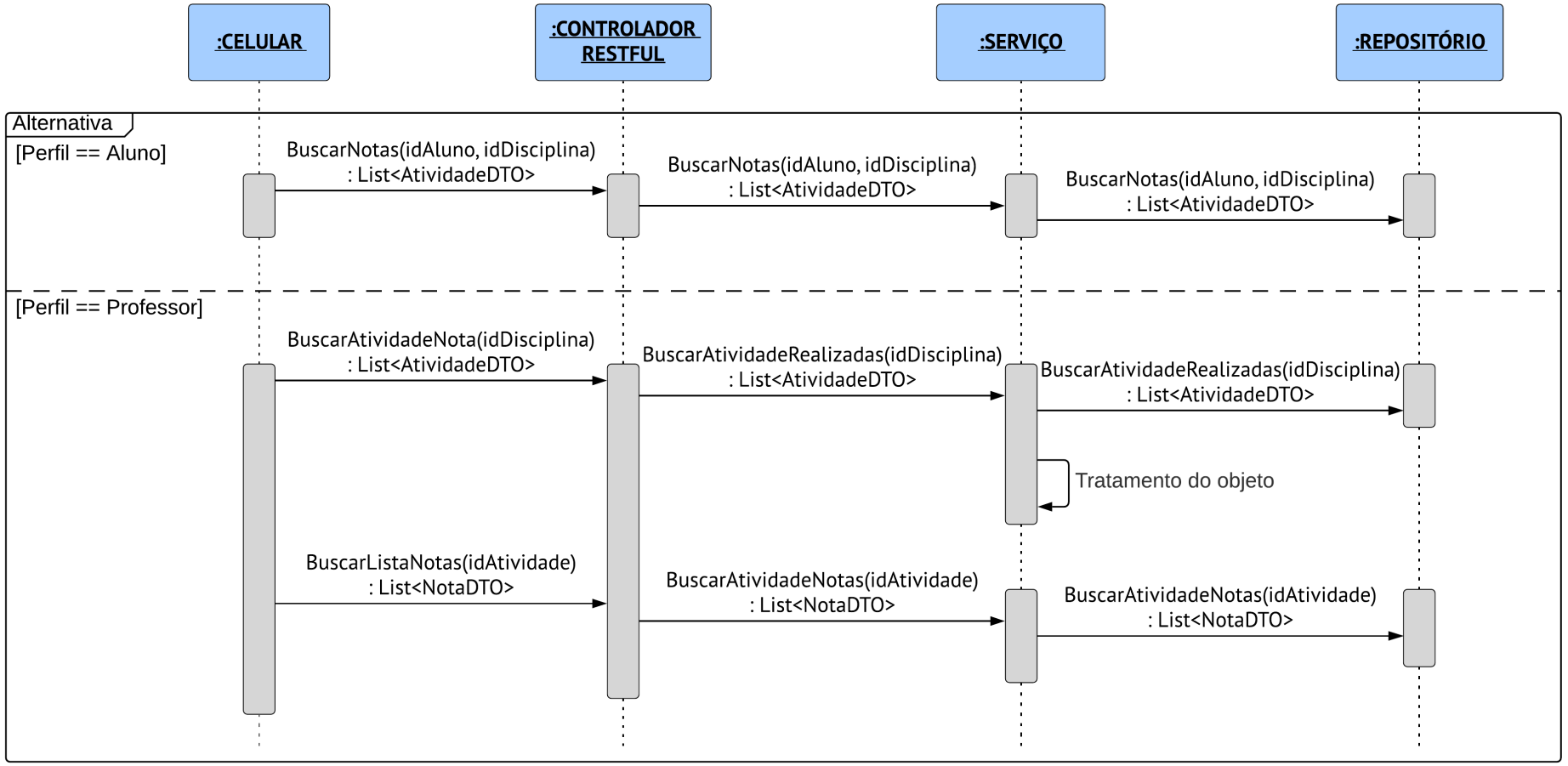
Já para o *Professor,* o retorno do método `BuscarAtividadeNota()`é uma lista com todas as atividades da disciplina. Quando um item da lista é selecionado, o método `BuscarNotas()` disparado retorna outra lista com todos *Alunos* cadastrados na disciplina e suas respectivas notas.

Figura 4.3 - Diagrama de Sequência UML para o requisito funcional RF14.



Fonte: Elaborado pelo autor.

Figura 4.4 - Diagrama de Sequência UML para o requisito funcional RF09.



Fonte: Elaborado pelo autor.

5. IMPLEMENTAÇÃO

Este capítulo discute a implementação do RF de código RF14 (Realizar chamadas, vide Quadro 4.2). Implementamos diversos outros RFs neste trabalho, mas de forma geral, podemos considerar que todas as funcionalidades do Akademi foram implementadas seguindo os mesmos passos dos códigos exibidos neste capítulo.

5.1 Requisição da API via Aplicativo

Nos exemplos a seguir, optamos por desconsiderar trechos de código da interface gráfica do aplicativo para facilitar o entendimento, concentrando na comunicação entre o Aplicativo e a API *web*. Em outras palavras, o código, escrito na linguagem de programação C#, presente no Quadro 5.1, parte do ponto em que o módulo aplicativo realiza a requisição para o módulo servidor.

Para a efetuar a requisição à API *web*, uma lista de chamadas que contém os alunos e se eles estavam presentes ou não na aula, é convertida para um objeto JSON (Linhas 04-06) e enviada por meio de uma instância de serviço HTTP usando o método `PostAsync` que recebe como parâmetro um caminho para API e nosso objeto JSON convertido (Linha 08). Caso essa requisição sofra alguma falha, o método `SalvarChamada()` retorna falso senão, verdadeiro (Linha 09-12).

Quadro 5.1 - Código do método `SalvarChamada` do módulo aplicativo do Akademi.

```
01  async Task<bool> SalvarChamada(ListaChamadasDTO listaChamadasDTO)
02  {
03    try {
04      var content = new StringContent(
05      JsonConvert.SerializeObject(listaChamadasDTO),
06                                  Encoding.UTF8, "application/json");
07      using (var response = await
08                _HttpClient.PostAsync("Disciplina/Chamada", content)) {
09        if (!response.IsSuccessStatusCode)
10          return false;
11
12        return true;
13      }
14    }
15    catch (Exception ex){
16      throw ex;
17    }
18  }
```

Fonte: Elaborado pelo autor.

5.2 Controlador

No Quadro 5.2, assim que a lista de chamada chega ao servidor via protocolo HTTP por uma solicitação POST (Linha 01) essa solicitação é autorizada para usuários com perfil “professor” (Linha 02).

Uma vez autorizada a requisição, a lista de chamadas é encapsulada em um objeto DTO (argumento `ListaChamadasDTO` no método `IActionReult`, Linha 03). De posse da lista de chamadas encapsulada em um objeto, o fluxo de controle é delegado para o método `Inserir` do componente de serviço (objeto `_chamadaService`, presente na Linha 07).

Caso não exista erros ao persistir a lista de chamadas no banco de dados, uma resposta de sucesso é retornada para o código cliente que fez a requisição (Linha 09). O método `Inserir` pode gerar uma exceção caso algum problema ocorra durante o processamento da lista de chamadas. Caso isso aconteça, uma resposta indicativa do erro é retornada à requisição HTTP (Linhas 11-13).

Quadro 5.2 - Código do método `InserirChamada` do controlador da API.

```
01  [HttpPost("Chamada")]
02  [Authorize(Roles = "professor")]
03  public IActionResult InserirChamada(ListaChamadasDTO dados)
04  {
05    try {
06      if (dados != null)
07        _chamadaService.Inserir(dados);
08
09      return Ok("Registro gravado com Sucesso");
10    }
11    catch (Exception e) {
12      return BadRequest(e.ToString());
13    }
14  }
```

Fonte: Elaborado pelo autor.

5.3 Serviço

O componente de serviço (referenciado por `_chamadaService` na Linha 07 do Quadro 5.2) recebe o fluxo de execução do Controlador e converte a lista de chamadas para um objeto específico para persistência no banco de dados, como podemos ver no Quadro 5.3 (Linhas 04-09). A persistência é feita delegando o fluxo

de execução mais uma vez, desta vez para o componente *Repositório* por meio de chamada ao método Inserir do objeto `_chamadaRepository` (Linha 10).

Quadro 5.3 - Código do método `Inserir` da camada de serviço.

```
01  public void Inserir(ListaChamadasDTO listaChamada)
02  {
03    foreach (var presenca in listaChamada.Presencas) {
04      Chamada chamada = new Chamada() {
05        DisciplinaId = listaChamada.DisciplinaId,
06        Data = listaChamada.Data,
07        AlunoId = presenca.AlunoId,
08        Presente = presenca.Presente
09      };
10    _chamadaRepository.Inserir(chamada);
11  }
```

Fonte: Elaborado pelo autor.

5.4 Repositório

Por fim, o método `Inserir()`, herdado de uma classe base BaseRepository que todos os repositórios do Akademi implementam, realiza a gravação dos dados da chamada no banco de dados e finaliza salvando as mudanças realizadas (Linhas 03-04).

Quadro 5.4 - Código do método inserir da camada repositório.

```
01  public void Inserir(T objeto)
02  {
03    bancoDados.Add(objeto);
04    bancoDados.SaveChanges();
05  }
```

Fonte: Elaborado pelo autor.

6. DISCUSSÕES, CONCLUSÕES E TRABALHOS FUTUROS

Apresentamos nesse trabalho a criação de uma aplicação distribuída para gestão acadêmica de alunos e professores. Imaginamos o Akademi como solução complementar aos sistemas de gestão de instituições públicas e particulares.

A vantagem de se utilizar o Akademi como solução complementar reside no fato de não ser necessária a integração total com os sistemas de gestão acadêmica presentes nas instituições, permitindo assim que os gestores avaliem ao longo do tempo a necessidade ou não de integração, substituição do sistema existente, ou apenas uso de maneira complementar.

Consideramos limitações deste trabalho a ausência de testes de experiência de uso com o usuário e ausência de testes de desempenho.

Esperamos que, no futuro, possamos disponibilizar o Akademi para dispositivos IOS, uma vez que já se encontra desenvolvido, porém não publicado pelas questões já abordadas na Seção 2.1.

E também criar um projeto *web* que seja integrado ao Akademi, trazendo independência da integração com outros sistemas, que hoje é necessário pelo fato de que não é possível, pelo aplicativo, cadastrar novas disciplinas, professores e alunos, novas notas e etc.

Por fim, há intenção de acrescentar à aplicação funcionalidades como, anexação de arquivos para o maior detalhamento de atividades criadas pelo professor nas disciplinas, possibilidade de alteração de foto de perfil dos usuários. E algumas melhorias como, emprego de um serviço de mensagens que poderá enviar notificações para os celulares de alunos e professores sem que seja necessário estarem com o aplicativo Akademi aberto.

7. REFERÊNCIAS BIBLIOGRÁFICAS

ESCOLA EM MOVIMENTO. Disponível em: <http://escolaemmovimento.com.br/>. Acesso em: 26 de março de 2018.

ESCOLA WEB. Disponível em: <https://escolaweb.com.br/>. Acesso em: 16 de dezembro de 2020.

GOADRICH, M. H. e ROGERS, M. P. (2011). ***Smart smartphone development: ios versus android***. Em *Proceedings of the 42nd ACM technical symposium on Computer Science education*, SIGCSE ’11, páginas 607–612, Nova York, NY, EUA. ACM.

GOOGLE. **Gerencie o ensino e a aprendizagem com o Sala de Aula**, 2020. Disponível em: <https://edu.google.com/intl/pt-BR/products/classroom>. Acesso em: 04 de dezembro de 2020.

LARMAN, Craig; **Utilizando UML e Padrões: Uma Introdução à Análise e ao Projeto Orientados a Objetos e ao Desenvolvimento Iterativo**, 3 ed. Bookman, 2007.

MICROSOFT. **Introdução ao ASP.NET Framework**, 2020. Disponível em: <https://docs.microsoft.com/pt-br/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-5.0>. Acesso em: 10 de novembro de 2020a.

MICROSOFT. **Documentação Técnica do SQL SERVER**, 2020. Disponível em: <https://docs.microsoft.com/pt-br/sql/sql-server/?view=sql-server-ver15>. Acesso em: 10 de novembro de 2020b.

MICROSOFT. **O que é Xamarin.Forms?**, 2020. Disponível em: <https://docs.microsoft.com/pt-br/xamarin/get-started/what-is-xamarin-forms>. Acesso em: 10 de novembro de 2020c.

MICROSOFT. **Bem-Vindo ao IDE Visual Studio**, 2020. Disponível em: <https://docs.microsoft.com/pt-br/visualstudio/get-started/visual-studio-ide?view=vs-2019>. Acesso em: 10 de novembro de 2020d.

MICROSOFT. **Criar APIs Web com ASP.NET Core**, 2020. Disponível em: <https://docs.microsoft.com/pt-br/aspnet/core/web-api/?view=aspnetcore-5.0>. Acesso em: 10 de novembro de 2020e.

MICROSOFT. **O que é o SSMS (SQL Server Management Studio)?**, 2020. Disponível em: <https://docs.microsoft.com/pt-br/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver15>. Acesso em: 10 de novembro de 2020f.

MICROSOFT. **Entity Framework Core**, 2020. Disponível em: <https://docs.microsoft.com/pt-br/ef/core/>. Acesso em: 16 de novembro 2020g.

MOODLE. **About Moodle**, 2020. Disponível em: <https://moodle.org/about/>. Acesso em: 10 de novembro de 2020.

PEREIRA, Lucio Camilo Oliva; DA SILVA, Michel Lourenço. **Android para desenvolvedores**. Brasport, 2009.

SILVIA, M. M. DA; SANTOS, M. T. P. **Os Paradigmas de Desenvolvimento de Aplicativos para Aparelhos Celulares**. T.I.S – Tecnologias, Infraestrutura e Software – UFSCar, v. 3, n. 2, p. 162 – 170, 2014.

SISTEMA ACADÊMICO UNIFAL. **Sistema Acadêmico**, 2020. Disponível em: <https://www.unifal-mg.edu.br/cienciassociais/sitemaacademico>. Acesso em: 10 de novembro de 2020.

SOLUTECH. **Acadêmico Online**, 2020. Disponível em: <http://www.solutechsistemas.com.br/produtos/academico-online>. Acesso em: 15 de setembro de 2020.

SOMMERVILLE, Ian. **Engenharia de Software**, 9 ed. São Paulo: Pearson Prentice Hall, 2011.

TIUWEB. **Terminal Informativo UNIFENAS**, 2020. Disponível em: <https://tiu.unifenas.br/>. Acesso em: 10 de novembro de 2020.

TRIGO, Antonio & HENRIQUES, Jorge. (2018). **Aprenda a programar com C#**. Livro de programação. Acesso em: 15 de setembro de 2020.

UFLA. **APLICATIVO MINHA UFLA GANHA NOVA VERSÃO**, 2017. Disponível em: <http://www.ufla.br/dcom/2017/06/28/aplicativo-minha-ufla-ganha-nova-versao/>. Acesso em: 10 de novembro de 2020.

Anexo A - Especificação de Requisitos

O Quadro A.1 descreve os RFs do aplicativo Akademi. A coluna “Objetivo” está atrelada aos códigos do objetivos descritos no Seção 4.2.

Quadro A.1 - Requisitos funcionais do Akademi.

Objetivo	Código	Requisito	Descrição
OBJ01	RF01	Visualizar aulas do dia	Deve ser possível que o aluno veja as aulas e horários que terá no dia. Assim como o professor verá as aulas a serem ministradas.
OBJ01	RF02	Visualizar atividades da disciplina	Deve ser possível visualizar todas as atividades de uma disciplina.
OBJ01	RF03	Visualizar próximas atividades	Deve ser possível visualizar as próximas atividades do professor e do aluno.
OBJ05	RF04	Cadastrar atividades	Deve ser possível para o professor cadastrar novas atividades nas disciplinas que ministra.
OBJ05	RF05	Editar atividades	Deve ser possível para o professor editar as atividades nas disciplinas que ministra.
OBJ05	RF06	Excluir atividades	Deve ser possível para o professor excluir atividades das disciplinas que ministra.
OBJ03	RF07	Visualizar itinerário de ônibus	Deve ser possível que o aluno veja os horários e destinos dos ônibus relacionados a sua instituição.
OBJ01	RF08	Listar disciplinas do professor e do aluno	Deve ser possível que o aluno veja as disciplinas em que está matriculado. Assim como o professor verá as disciplinas que ministra.
OBJ02	RF09	Visualizar notas	Deve ser possível que o aluno veja suas notas nas disciplinas. Assim como o professor verá as notas de seus alunos em cada atividade das disciplinas que ministra.
OBJ05	RF10	Alterar notas	Deve ser possível para o professor alterar a nota da atividade de algum aluno.

Quadro A.1 - Requisitos funcionais do Akademi.

OBJ01 e OBJ04	RF11	Enviar mensagens	Deve ser possível que o aluno e professor envie mensagens em um *chat* da disciplina.
OBJ01 e OBJ04	RF12	Receber mensagens	Deve ser possível que o aluno e professor receba mensagens em um *chat* da disciplina.
OBJ02	RF13	Visualizar frequências	Deve ser possível que o aluno veja sua frequência em cada disciplina matriculado.
OBJ05	RF14	Realizar chamadas	Deve ser possível que professor realize chamada nas aulas.
OBJ05	RF15	Alterar presenças	Deve ser possível que professor altere a presença de uma chamada existente.
OBJ06	RF16	Cadastrar tarefa semanal	Deve ser possível para o professor cadastrar tarefas semanais para melhorar sua organização.
OBJ06	RF17	Excluir tarefa semanal	Deve ser possível para o professor excluir tarefas semanais.
OBJ01	RF18	Visualizar perfil de membros	Deve ser possível para o aluno visualizar o perfil dos membros das disciplinas em que está matriculado.
OBJ01	RF19	Receber aviso de aula	Deve ser possível para o aluno visualizar o aviso das aulas do dia. Esse aviso será criado pelo professor.
OBJ05	RF20	Criar aviso de aula	Deve ser possível para o professor notificar os alunos com algum aviso nas suas aulas do dia.
OBJ01 e OBJ05	RF21	Login no aplicativo	Para conectar no aplicativo deve ser informado e-mail e senha do aluno ou professor.
OBJ01 e OBJ05	RF22	Logout do aplicativo	Para desconectar do aplicativo o aluno ou professor deve clicar em um botão na tela principal.

Fonte: Elaborado pelo autor.

Anexo B - Telas do Aplicativo

Neste anexo estão as telas criadas para o Akademi. As Figuras B.1 e B.2 apresentam o menu principal do professor e do aluno respectivamente.

No menu principal do professor e aluno, encontramos as disciplinas com aulas previstas para o dia corrente (parte superior com fundo verde). Para os alunos é apresentado os horários referentes ao transporte que os leva até a instituição de ensino (parte inferior com fundo cinza) e, na mesma parte, aos professores são apresentadas tarefas semanais que podem ser cadastradas por eles.

Outra informação que pode aparecer em ambos atores é um aviso rápido enviado pelo professor referente a sua aula corrente (caixa vermelha com fundo verde na Figura B.1)

Figura B.1 – Menu principal Professor.

Figura B.2 – Menu principal Aluno.

A Figura B.3 apresenta a lista de disciplinas do aluno (quadros com bordas azuis) em que, na parte esquerda da tela encontramos os nomes das disciplinas e embaixo seu respectivo professor. Ao lado direito da tela temos a média ponderada do aluno na disciplina até o momento atual.

A diferença da tela de disciplinas do aluno com a do professor é que a deste não exibe as médias e não apresenta o seu nome abaixo do nome da disciplina.

Figura B.3 – Tela de Disciplinas.

A Figura B.4 nos mostra a última opção do menu inferior do Akademi que é “Atividades" e, nesse caso, semelhante para os dois atores.

Tal opção exibe descrição da tarefa, data que irá acontecer e nome da disciplina, das tarefas futuras de todas disciplinas que o professor ou o aluno está relacionado.

Figura B.4 – Tela de Atividades.

A Figura B.5 apresenta uma lista de atividades com nome da tarefa, tipo e peso, que já tiveram suas notas gravadas no banco de dados.

Selecionando algum desses itens, uma lista com notas lançadas de cada aluno é exibida (podemos ver essa lista ao fundo da Figura B.6 com tom mais escuro). Então, ao selecionar algum registro de nota do aluno é apresentada uma tela para alteração de nota (parte central com fundo branco da Figura B.6).

Figura B.5 – Tela de Atividades com Nota do Professor.

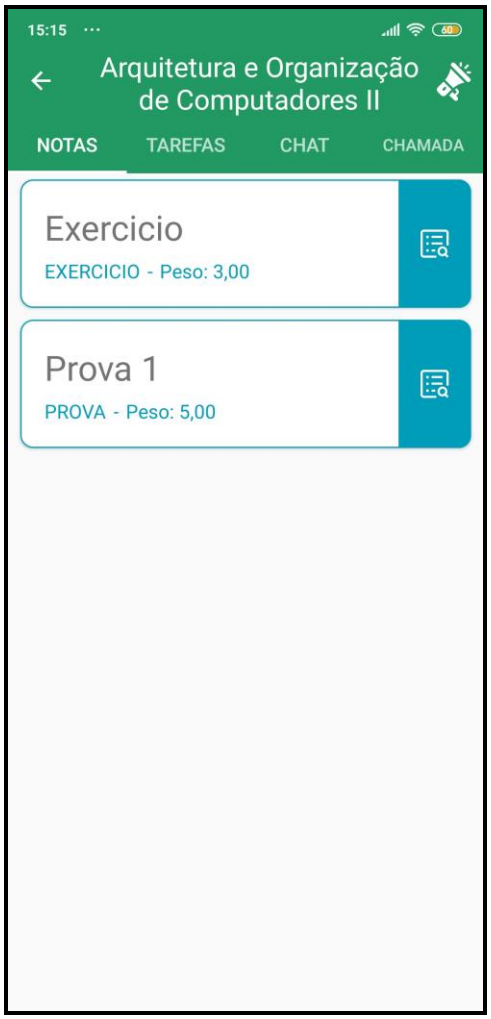
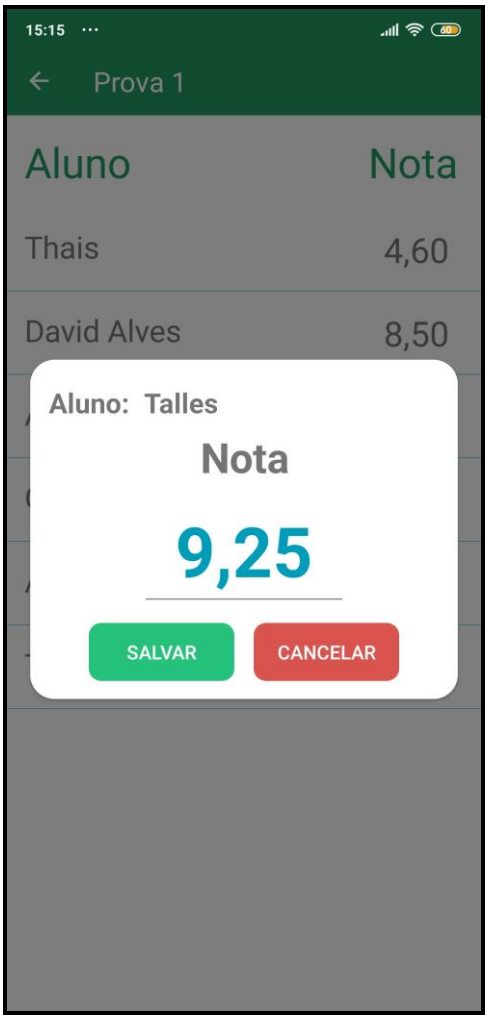
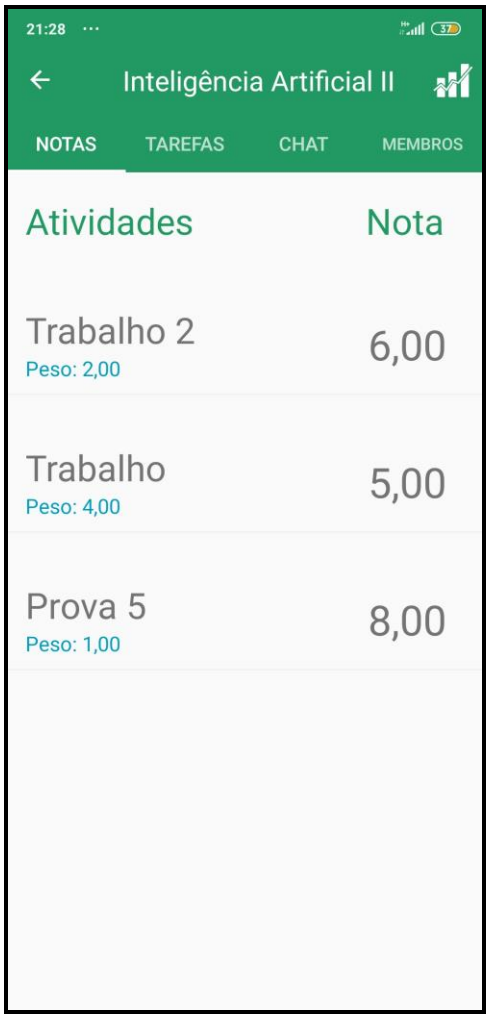


Figura B.6 – Tela de Alteração de Notas do Professor.



A Figura B.7 apresenta uma lista de atividades com notas já lançadas do aluno para uma determinada disciplina, trazendo as informações de nome da atividade, peso e nota atingida.

Figura B.7 – Tela de Notas do Aluno.



A Figura B.8 apresenta os horários de início e término das aulas durante a semana de todas disciplinas relacionadas ao aluno ou professor.

Figura B.8 – Tela de Horários de Aula Semanais.

As Figuras B.9 e B.10 estão relacionadas às tarefas de uma disciplina. Ao acessar a aba de Tarefas de um disciplina, uma lista de tarefas é carregada semelhante a tela do menu de atividades (Figura B.4), porém exibe apenas as tarefas de uma disciplina e não todas.

Tanto aluno ou professor possuem a aba de tarefas e ao selecionarem algum item da lista são direcionados à tela da Figura B.10 que exibe as informações da tarefa.

A diferença entre o aluno e o professor quando acessarem a tela da Figura B.9 e B.10 será que as opções de inclusão, alteração e exclusão (ícones na parte inferior direita com fundo claro e parte superior direita com fundo verde) não serão exibidas para o aluno.

Figura B.9 – Tela de Tarefas do Professor.

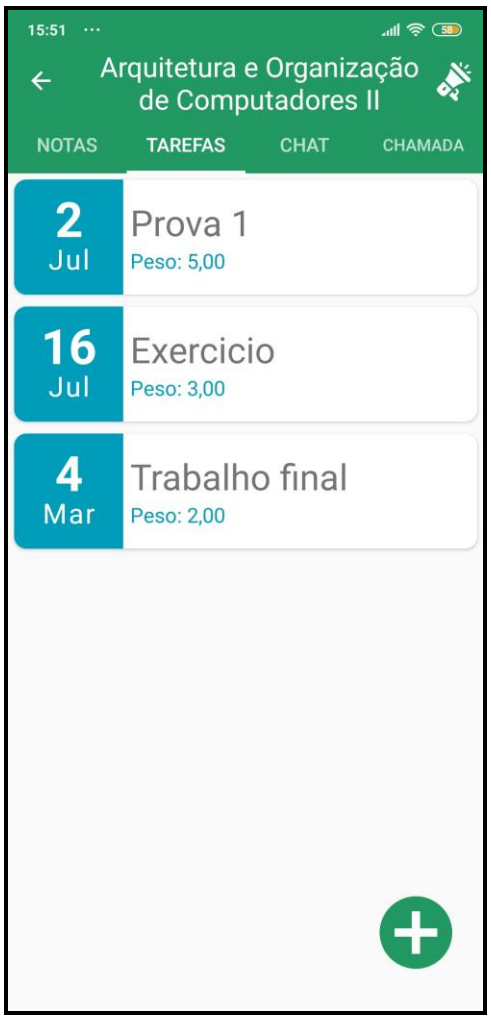
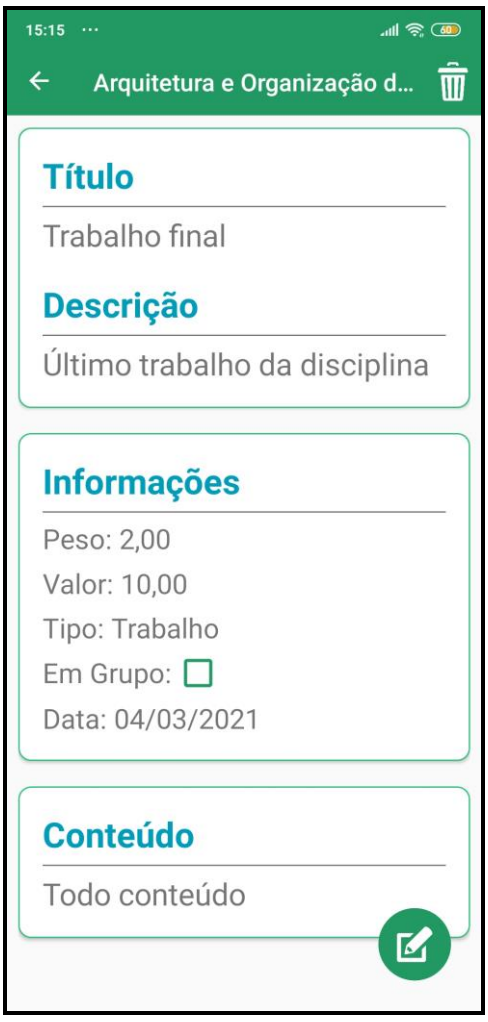
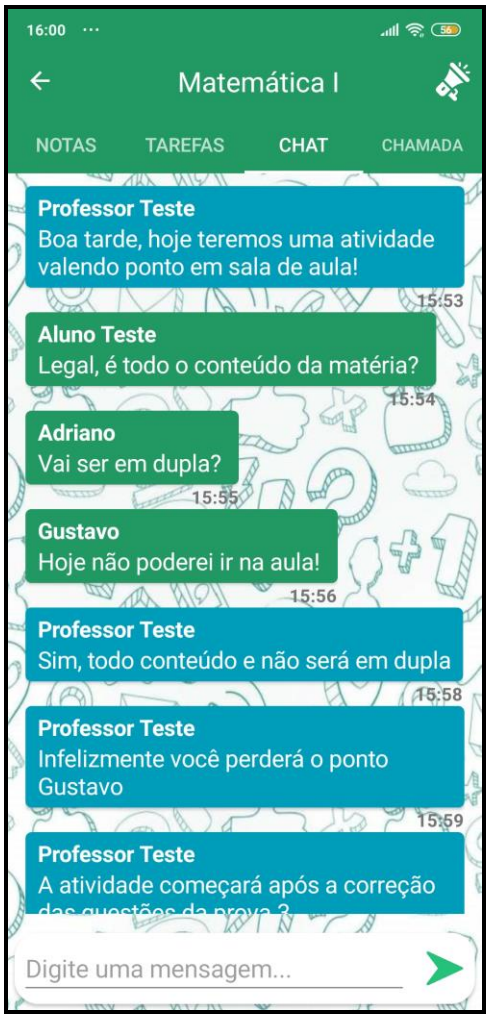


Figura B.10 – Tela de Informações da Tarefa do Professor.



A Figura B.11 apresenta o chat da disciplina em que professor e alunos relacionados podem trocar mensagens.

Figura B.11 – Chat da Disciplina.



As Figuras B.12 e B.13 apresentam uma lista com todos os alunos matriculados na disciplina e as informações cadastradas de cada aluno, respectivamente.

Figura B.12 – Tela de Membros da Disciplina.

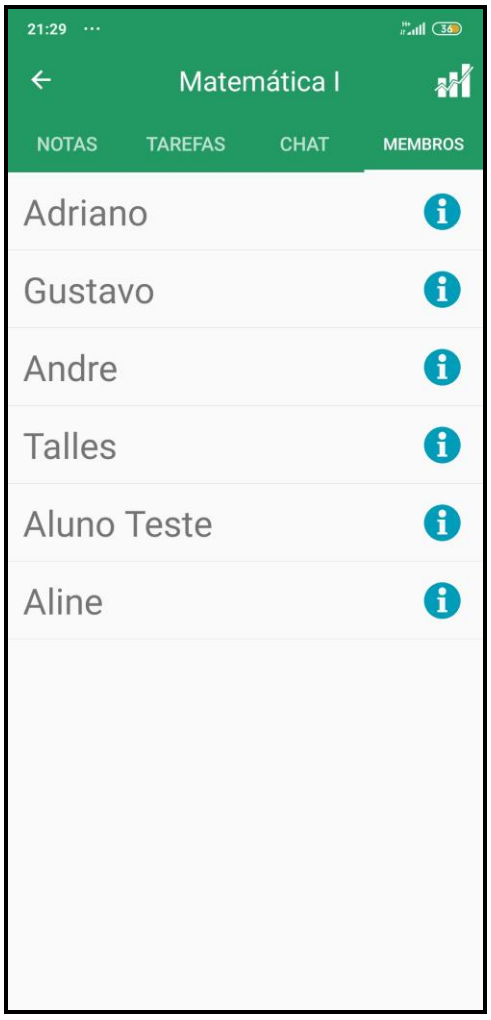
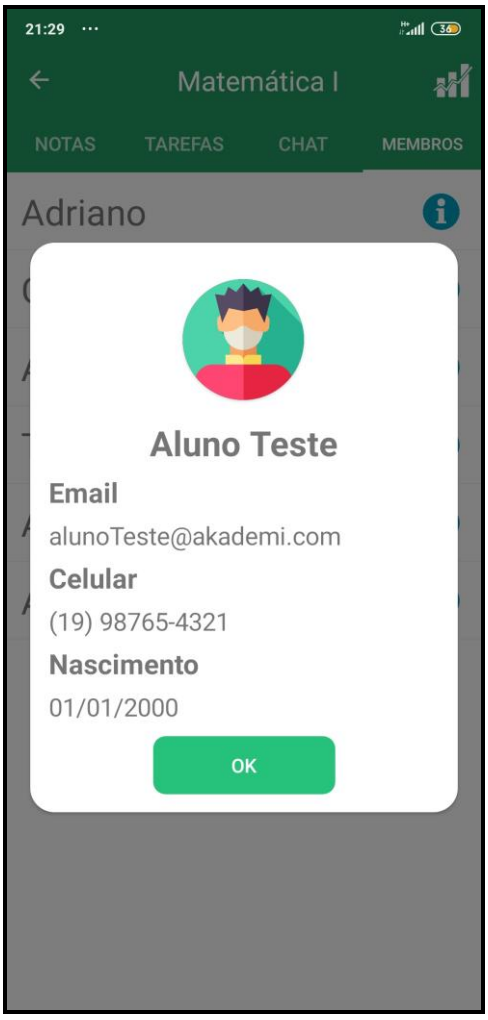


Figura B.13 – Tela de Informações do Membro da Disciplina.



Ambas as telas apresentadas nas Figuras B.14 e B.15 são de acesso exclusivo do professor relacionado a disciplina por onde estão sendo acessadas.

Na Figura B.14 temos uma lista de todas as chamadas realizadas na disciplina que, selecionando alguma delas, o professor poderá fazer alteração da presença do aluno caso por descuido tenha gravado errado.

A Figura B.15 exibe a tela de cadastro de chamadas da disciplina em que o professor informará quais alunos estiveram presentes ou não, bem como a data da chamada.

Para acessar a tela de cadastro de chamadas é necessário selecionar o ícone de chamada (ícone na parte superior direita com fundo verde da Figura B.14).

Figura B.14 – Tela de Chamadas da Disciplina.

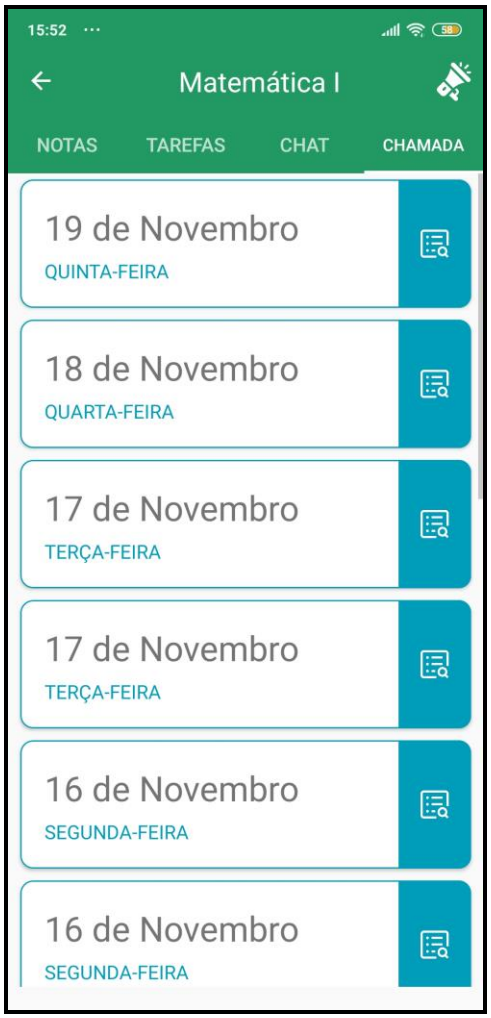
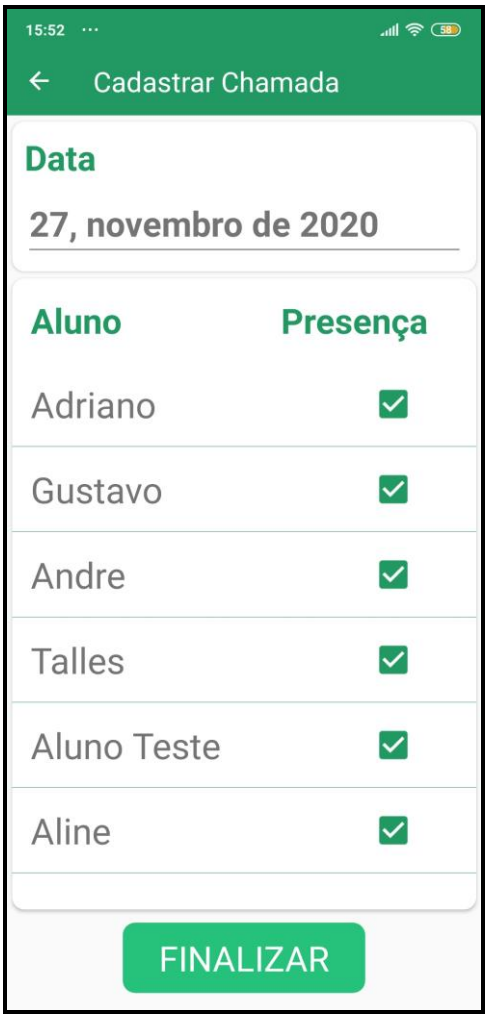


Figura B.15 – Tela de Cadastro de Chamada.



A Figura B.16 apresenta uma tela em que apenas os alunos têm acesso, que exibe uma análise da frequência do aluno em uma disciplina.

Na parte superior da figura temos o gráfico de frequência do aluno que altera sua cor de acordo com a porcentagem de presença.

Mais abaixo temos uma contagem de presenças e faltas e, na parte inferior abaixo da contagem, uma lista com a data de todas as faltas do aluno.

Figura B.16 – Frequência do Aluno