ADA - Atribuição de Aulas

Ana Paula Moura Messias de Souza, Gustavo Santos Costa Soares, Henrique Luis Baesa, Isabella Valerio Mazará, Josineudo das Chagas Arruda, Paulo Kenji Yokota Muneischi

[1]Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP)
São Paulo – SP – Brasil

[2]Curso Técnico em Informática Integrado ao Ensino Médio
PDS - Pratica para Desenvolvimento de Sistemas

ada.ifsp@gmail.com

moura.a,santos.soares,h.baesa,i.mazara,josineudo.c, paulo.muneischi

@aluno.ifsp.edu.br

Abstract. *This document describes the final delivery of the Assignment of Classes (ADA) web application project, developed in the Practical for Systems Development (PSD) course at the Federal Institute of Education, Science and Technology - São Paulo Campus. The ADA aims to automate the aforementioned attribution system, in order to provide ease and agility to the employees involved. Therefore, the target audience are professors and other positions responsible for the general administration and for the representation of each block when configuring the necessary details in the execution of the processes. Keywords: assignement of classes, automation, IFSP-SPO, teachers, regulations, FPA.*

Resumo. *Este documento descreve a entrega final do projeto da aplicação web* ***Atribuição de Aulas (ADA)****, desenvolvido na disciplina Prática para Desenvolvimento de Sistemas (PDS) no Instituto Federal de Educação, Ciência e Tecnologia - Câmpus São Paulo (IFSP-SPO). A ADA visa automatizar o referido sistema de atribuição do instituto, de modo a proporcionar facilidade e agilidade aos funcionários envolvidos. Logo, o público-alvo são os docentes e os demais cargos responsáveis pela administração geral e pela representação de cada bloco ao configurarem os detalhes necessários na execução dos processos. Palavras-chave: atribuição de aulas, automatização, IFSP-SPO, docentes, regulamentos, FPA.*

1. Introdução

O sistema de atribuição do IFSP-SPO envolve um processo complexo, porém essencial, para a designação dos docentes em suas futuras aulas, garantindo o cumprimento de sua carga horária e, por conseguinte, a realização efetiva de suas responsabilidades. É conduzido semestralmente, a cada seis meses, com exceção das disciplinas de cursos integrados e PROEJA[1], que ocorrem a cada dois semestres. E, desde 2021, o processo leva cerca de dois meses para ser efetuado e efetivado por completo.

[1]Programa Nacional de Integração da Educação Profissional com a Educação Básica na Modalidade de Educação de Jovens e Adultos

Para isso, é seguido o regulamento vigente, a Portaria nº SPO.071, de 15 de março de 2023, que abarca todas as orientações e está embasada na Resolução IFSP nº 112/2014, de 07 de outubro de 2014, alterada posteriormente pela Resolução IFSP nº 109/2015, de 04 de novembro de 2015. Além de serem seguidos o Parecer nº 146/2015/CONSUL/PEIFSÃOPAULO/PGF/AGU e os da CNE/CEB[2], em específico a diretriz nº 5/1997, aprovada em 7 de maio de 1997, responsável pela regulamentação da Lei nº 9.394/96.

Devido ao seu elevado grau de complexidade, diversos profissionais estão envolvidos, são eles o Diretor-Geral, os diretores de departamento, os coordenadores de curso superior, os representantes de subárea, os docentes, e os diretores de departamento, de ensino e de pesquisa, extensão e pós-graduação. Ademais, os órgãos envolvidos são a Coordenadoria de Turnos (CTU), a Diretoria de Ensino (DEN) e a Diretoria de Pesquisa, Extensão e Pós-graduação (DPE)[3].

Atualmente, para a execução do sistema, o processo é executado através de documentos estruturados manualmente e disponibilizados no ambiente virtual eadcampus; a atribuição, em particular, através de múltiplas planilhas detalhadas do Excel. E, na resolução dos conflitos decorrentes de insatisfações individuais, escassez de aulas aos docentes remanescentes, divergências de horários e afins, são realizadas negociações extraoficiais.

Ciente da complexidade, a ADA visa automatizar a atribuição, em específico os procedimentos do Art. 3º da Portaria, parágrafos I, II, III, IV, V, VI, VIII, IX e XI. Isso abrange o estabelecimento dos prazos a serem cumpridos; a configuração do ambiente com os dados necessários sobre os horários, as turmas, as disciplinas e os usuários; a determinação do critério de prioridade dos docentes para a atribuição; a realização e entrega do Formulário de Preferência de Atividades (FPA)[4]; a atribuição automática e a manual, em caso de conflitos de preferência; e a possibilidade de permuta após a atribuição, se desejado e viável.

Nesse contexto, evidentemente, a aplicação *web* considera o regime de trabalho do docente e assegura o cumprimento das leis trabalhistas e dos regulamentos da Instituição. Essas englobam o número máximo de 25 alunos por turma, uma jornada diária não superior a 8 horas, e o respeito à interjornada de 11 horas. Igualmente, aplica as especificidades do cumprimento de regência de aulas mínimo e máximo de cada regime (20 ou 40 horas).

1.1. Justificativa

O sistema atual de atribuição demonstra eficácia, entretanto, carece de eficiência, evidenciando várias oportunidades de aprimoramento. No referido Instituto, há mais de 350 docentes, destes, 56 concentrados no Departamento de Informática e Turismo (DIT) – a área onde a ADA foi criada. Todos participam do sistema de atribuição semestral, submetendo os documentos requeridos e passando por esse procedimento. O resultado é uma considerável quantidade de dados a serem avaliados e de pessoas a serem gerenciadas.

Consequentemente, a organização é comprometida, ocasionando desafios na

[2]Câmara de Educação Básica do Conselho Nacional de Educação.

[3]As atribuições e responsabilidades desses envolvidos são descritas na Portaria, juntamente com todos os procedimentos necessários, localizada no anexo –.

[4]Preenche com seus dados pessoais, disponibilidade de horário, prioridades de aulas, atividades de apoio ao ensino e complementação de atividades (modelo no anexo –).

execução dos procedimentos e insatisfações no desenvolvimento e nos resultados do processo; pela parte dos docentes e dos diretores, coordenadores e representantes. Principalmente nos procedimentos que envolvem o FPA, a atribuição e a possibilidade de permuta, são frequentes os relatos referentes a equívocos no preenchimento, atrasos na submissão, excesso de trabalho manual na atribuição, decisões e alterações em estágios avançados, além de conflitos interpessoais gerados e a decorrente necessidade de comunicações particulares constantes entre o representante do bloco e os docentes[5]. Isso foi atestado por meio de uma pesquisa oral com os representantes da DIT, André Evandro Lourenço e Leonardo Andrade Motta de Lima, e com os docentes pertencentes a todos os blocos.

Em adição, a quantidade de procedimentos exigidos é considerável, exigindo a troca constante de documentos entre os departamentos e a utilização de múltiplos canais de comunicação[6], o que prejudica o controle eficiente do que foi feito (ou não), por quem e em que momento, especialmente caso seja necessária a comprovação das ações.

Em suma, esses fatores, em conjunto com a intrincada organização necessária para geri-los, tornam o sistema de atribuição exaustivo e contencioso a todos os envolvidos, demandando um tempo e recursos para armazenamento de dados anteriores que poderiam ser reduzidos significativamente.

1.2. Objetivo

A ADA é uma aplicação *web* que visa aprimorar não apenas a eficácia, mas, também, a eficiência do sistema de atribuição do DIT. Com essa perspectiva, tem como principal objetivo facilitar a vida dos profissionais envolvidos, a partir da redução do tempo de execução do processo e da aprimoração da gestão dos dados e dos usuários, enquanto promove a transparência.

Ao proporcionar um melhor desempenho, almeja aprimorar a organização, possibilitando o acompanhamento mais eficiente dos procedimentos. E, especificamente nos casos de relatos de erros frequentes, a ADA busca implementar os regulamentos de forma mais confiável, estabelecendo uma padronização e restrição às configurações da atribuição. Inclusive, busca reduzir interferências humanas desnecessárias e repetitivas, assim como, reduzir a necessidade de comunicações extraoficiais, com a resolução de conflitos de preferências diretamente através da aplicação.

Além disso, a aplicação pretende tornar a quantidade de procedimentos mais enxuta, por meio da centralização dos principais em um só local, a ADA. Isso permitirá o controle das ações mais eficiente, por garantir de transparência na monitorização tanto do desempenho dos usuários quanto dos procedimentos, e, igualmente, por facilitar a geração de relatórios com dados, comprovações e auditorias. Como adicional, a integração do banco de dados tornará o acesso e o armazenamento de configurações anteriores mais intuitivas e leves.

Assim, uma operação que outrora dependia de organização manual e experenciava desafios no gerenciamento, muitas vezes mais suscetível a erros devido à subjetividade humana, será uma operação tecnológica mais limpa e funcional, onde a interferência humana será restrita a funções indispensáveis.

[5]Realizadas por e-mail, mensagens e/ou ligações.

[6]O comunicador Sistema Unificado de Administração Pública (SUAP), o ambiente virtual eadcampus e o e-mail institucional.

2. Análise de Concorrência

A análise do mercado de sistemas de atribuição automatizados resultou na identificação de uma aplicação principal, desenvolvida pela Secretaria da Educação do Estado de São Paulo (SED), e de uma aplicação secundária, pelo Sistema Integrado de Gestão (SIG).

A da SED abrange toda a rede educacional estadual de São Paulo (SP), e, de acordo com seus registros (vide anexo –), atende a aproximadamente 6 mil docentes. Em resumo, os docentes selecionam a(s) escola(s) e disciplina(s) na(s) qual(is) desejam lecionar, indicando interesse em aulas vagas, livres ou de substituição – em casos de ausência ou afastamento. Essa aplicação se destacou pela sua complexidade e pelo desenvolvimento semelhante ao proposto pela ADA: a atribuição se baseia em uma fila, porém, o critério é estabelecido por um sistema de pontuação; a escolha das escolas é realizada de acordo com a prioridade definida pelo usuário; os diretores têm acesso a todo o processo.

Quanto a do SIG, abrange exclusivamente as Escolas Técnicas Estaduais (Etecs). E, por sua vez, essa aplicação foi considerada como secundária, dado que concentra-se no preenchimento e submissão de inscrição e requerimentos de ampliação de carga horária pelos docentes, entretanto, não realiza a atribuição automatizada em si[7].

A equipe da ADA prevê o tratamento de algumas questões presentes na aplicação da SED, como o gerenciamento da manipulação de múltiplas filas. Igualmente, reconhece outros potenciais pontos de atenção, como o atraso causado por longas filas de seleção; a ausência de verificação de componente curricular, que permite docentes sem o nível de escolaridade necessário lecionarem a disciplina; e os erros no processo de pontuação para prioridade de escolha. [6] [1]

Ao mesmo tempo também se reconhecem melhorias a serem implementadas, como abranger melhor todos os casos de exceção – como ausência ou afastamento de um docente –, e discutir a possibilidade de alteração da atribuição final pelo Representante ou Administrador. Em relação ao SIG, detalhes adicionais não puderam ser verificados devido à inacessibilidade ao seu tutorial. Porém, a página que permite aos docentes indicar sua disponibilidade de horário é clara (anexo –) e pode servir como um exemplo.

Com base no exposto, a ADA se destaca ao subsidiar uma série de ações que permitem a integração de dados e procedimentos de maneira inovadora, especialmente no âmbito de regulamentos do Instituto e na simplificação das fases no sistema de atribuição. E, dado seu público menor, oferece maior organização, análise de feedbacks e tratamento de erros, com a vantagem de proporcionar um contato direto com os usuários para esclarecimento de dúvidas, o bot Adaline e Adam.[8]

Portanto, a sua implementação considerará as melhores práticas das aplicações com propósitos similares, e, especialmente, as dificuldades que enfrentaram, não se atendo a um ciclo de falhas.

Por fim, é importante ressaltar que, por ambas as aplicações (da SED e do SIG) serem direcionadas a clientes específicos, como a ADA, não são consideradas concorrentes diretas, mas, podem servir como fonte de experiência e aprendizado; é mais apropriado considerá-las aplicações com objetivos similares em vez de concorrentes diretas.

[7] Informações disponibilizadas na página da Unidade de Recursos Humanos (URH), vide anexo –

[8] No apêndice A, há uma tabela comparando as aplicações para melhor visualização.

3. Revisão da Literatura

3.1. Instituição Federal

Em 1909, foi instaurada a Rede Federal de Educação Profissional, Científica e Tecnológica, com o Decreto nº 7.566, realizado pelo Presidente Nilo Peçanha em resposta à crise econômica e política da época. As instituições de educação profissional da rede se caracterizam por sua estrutura diferenciada, uma vez que foram criadas pela agregação/transformação de antigas instituições profissionais. [10]

O IFSP foi fundado nesse mesmo ano, como Escola de Aprendizes Artífices, e teve seu início efetivo em 1910. O Campus São Paulo foi uma das primeiras escolas, começando provisoriamente como um galpão na Avenida Tiradentes. [5] Transitou por diferentes processos e denominações ao longo dos anos, todavia, manteve o objetivo de oferecer uma formação de qualidade, com a condição de escola pública vinculada à União e, também, o prestígio junto à sociedade paulistana. [4]

Com o avanço industrial da década de 80, um novo cenário econômico e produtivo se estabeleceu, com o desenvolvimento de novas tecnologias, agregadas à produção e à prestação de serviços. Para atender a essa demanda, as instituições de educação profissional vêm buscando diversificar programas e cursos para elevar os níveis da qualidade da oferta. [8] Foi nessa época que o Instituto ampliou a sua atuação e seus objetivos oferecendo cursos superiores na Unidade Sede São Paulo. [5]

Posteriormente, foram implementados diversos cursos voltados à formação de tecnólogos na área da Indústria e de Serviços, Licenciaturas e Engenharias. [5] E, com a Lei nº 11.892, de 29 de dezembro de 2008, o IFSP ganha seu nome oficial, antes Cefet-SP, Centros Federais de Educação Profissional e Tecnológica.

Atualmente, o Instituto se localiza na Rua Pedro Vicente, 625, no Bairro do Canindé, além do desenvolvimento das atividades educacionais, abriga a sede da reitoria da instituição. [5] Sua estrutura organizacional acadêmica é composta por cinco departamentos, de Ciências e Matemática (DCM), de Construção Civil (DCC), de Elétrica (DEL), de Humanidades (DHU), de Informática e Turismo (DIT), e de Mecânica (DME). E, com eles, dedica-se a fornecer Cursos Técnicos (integrado ao médio, concomitante ou subsequente) e de curta duração (extensão), Graduação (Bacharelado, Licenciatura, Tecnologia e Educação à Distância) e Pós-Graduação (especialização e mestrado).Essas informações estão presentes no site do IFSP-SPO, na área Institucional/Cursos.

Os docentes são profissionais com experiência na educação básica, na docência superior e também no mercado de trabalho: aptos, portanto, a apresentar exemplos práticos de aplicação da teoria ministrada em diferentes grades curriculares. [7]

Como centro criador de ciência e tecnologia e com a vasta experiência e competência acumuladas em sua extensa trajetória, o IFSP tem capacidade para proporcionar aos seus estudantes uma visão crítica do conjunto do sistema e do processo produtivo e para contribuir com a educação brasileira, praticando a educação como efetivo fator de desenvolvimento humano e social. [5]

3.2. Regulamentos

A criação da Justiça do Trabalho, pelo Decreto nº 1.237, em 1939, e sua posterior regulamentação, em 1940, e instalação, em 1941, trouxe a necessidade do estabeleci-

mento da Consolidação das Leis de Trabalho (CLT). Por sua vez, essa apenas foi oficializada dois anos depois, pelo Decreto-Lei nº 5452, após a luta de juristas para promover a proteção do trabalhador de forma definitiva e unificar a legislação trabalhista já existente no Brasil.

É na legislação trabalhista que são estabelecidos os direitos e deveres de empregados e empregadores como, por exemplo, jornada de trabalho, remuneração, férias, aviso prévio, licenças, rescisão de contrato de trabalho, normas de segurança do trabalho e outras regras fundamentais para as relações de trabalho. [11]

Enquanto, no âmbito da educação, um grupo de educadores conceituados redigiu o Manifesto dos Pioneiros da Educação Nova, em 1932, para conseguir estabelecer um programa de política educacional amplo e integrado. O documento defendia: a educação como uma função essencialmente pública; a escola única e comum, sem privilégios econômicos de uma minoria; formação universitária para todos os professores; ensino laico, gratuito e obrigatório. [9]

Apenas no governo de Fernando Henrique Cardoso, em 1995, passou a ter atribuição exclusiva e começou a redigir leis para promover um ensino de qualidade, cumprindo com o proposto no Manifesto. Assim, tem como áreas de competência a política nacional de educação; a educação infantil; a educação em geral, compreendendo ensino fundamental, ensino médio, educação superior, educação de jovens e adultos, educação profissional e tecnológica, educação especial e educação a distância, exceto ensino militar; a avaliação, a informação e a pesquisa educacionais; a pesquisa e a extensão universitárias; o magistério e a assistência financeira a famílias carentes para a escolarização de seus filhos ou dependentes. [9]

No IFSP-SPO, os regulamentos trabalhistas e educacionais são aplicados em inúmeras situações, regem cada aspecto do Instituto – além dos seus próprios regulamentos. No entanto, no sistema de atribuição de aulas, há alguns específicos que influenciam diretamente na execução do processo.

Quanto a interjornada, o Artigo 66 da CLT determina: Entre 2 (duas) jornadas de trabalho haverá um período mínimo de 11 (onze) horas consecutivas para descanso. [3] Seu escopo principal é proporcional o sono ao trabalhador, ainda que se trate de pessoa submetida a jornadas alternadas ou diurnas. [2]

Tratando-se de docentes, exclusivamente, os Pareceres da CNE/CEB são cumpridos, em específico a diretriz nº 5/1997, aprovada em 7 de maio de 1997, responsável pela regulamentação da Lei nº 9.394/96. Nela, confirma-se a atribuição de, no máximo, 8 horas diárias, equivalentes a dez aulas de 45 minutos. Sobre o sistema de atribuição em si, semestralmente tem a divulgação de uma nova Portaria, com os procedimentos a serem prosseguidos na atribuição, os documentos a serem entregues (e em qual local), os cargos envolvidos e suas respectivas funções, além dos regulamentos e exceções a serem seguidos e considerados.

Por fim, até o momento, rege a Resolução IFSP nº 109/2015, a qual as Portarias se embasam. Nela, estão presentes as alterações no Regulamento de Atribuições de Atividades Docentes, com um maior detalhamento quanto a casos diversos e especificidades do regime de trabalho, das atividades docentes, da distribuição de carga horária, do processo de atribuição e das documentações exigidas para entrega.

3.3. Automatização

Uma característica definidora da espécie humana é a capacidade de inventar e refinar continuamente ferramentas que ajudam a aliviar os fardos do trabalho [12]. Desde muito antes da história registrada, os seres humanos vêm inventando maneiras de simplificar as atividades diárias para aumentar sua produtividade [12].

O termo ”automatização” foi criado na década de 1940, quando um engenheiro da Ford Motor Company o utilizou para descrever a maior mecanização das linhas de montagem de automóveis [13], o que proporcionou a diminuição do trabalho braçal. Entretanto, a automatização já existia há muito tempo, seu conceito esteve presente na pré-história, história antiga, início da era comum, renascimento, primeira revolução industrial e na segunda revolução industrial [12], onde o termo foi criado de fato. E atualmente, a automatização continua crescendo em todo o ecossistema de tecnologia de produção em todos os setores e em empresas de todos os tamanhos [14].

A Automatição de TI, às vezes referida como automatição de infraestrutura, é o uso de software para criar instruções e processos repetíveis para substituir ou reduzir a interação humana com sistemas de TI [18]. Ao aplicá-la em fluxos e processos de trabalho, principalmente aqueles que exigem esforço manual, é possível liberar e dedicar mais recursos a tarefas mais estratégicas [16]. Dessa forma, a automatização de um processo manual e repetitivo pode trazer uma maior produtividade e disponibilidade [17] aos funcionários.

A automatização executa o trabalho sozinha, não analisa o próprio desempenho e precisa de monitoramento constante. Enquanto a automação também executa o trabalho sozinha e precisa de monitoramento, mas é capaz de analisar o próprio trabalho em tempo real e tomar decisões. [19] No caso do IFSP-SPO, a primeira é preferível, pois proporciona maior flexibilidade, necessária pela constante mudança nos regulamentos e nas agendas e preferências dos docentes. Assim, essa natureza dinâmica exige uma solução que possa se adaptar prontamente a mudanças regulatórias, ajustes de cronograma e modificações nas preferências individuais dos professores.

Em vista disso, percebe-se que os seres humanos sempre percorreram um caminho para diminuir esforços; a ADA tem a finalidade de contribuir para esse propósito, simplificando e otimizando o processo de atribuição de aulas. Logo, liberta os envolvidos de tarefas repetitivas e permite que se concentrem em atividades de maior valor agregado.

Consequentemente, o seu desenvolvimento não apenas se alinha com a longa trajetória de automatização da humanidade, mas também representa um passo significativo na melhoria da eficiência e qualidade dos processos educacionais, com uma abordagem flexível que permite maior capacidade de personalização para que a ferramenta seja moldada de acordo com as especificidades do IFSP-SPO, resultando em um sistema de atribuição de aulas que se adapta ao invés de impor limitações.

4. Identidade

A Atribuição de Aulas[9], denominada ADA, teve sua sigla originada tanto da abreviação de seu nome quanto em homenagem à Ada Lovelace — renomada matemática, escritora

[9] A logo está no apêndice –

e considerada a primeira programadora da história[10].

Quanto à equipe, denominada Mottarios[11], se demonstrou ao longo do percurso autogerenciável e responsável, demonstrando proatividade em relação às exigências do projeto e dedicação plena à disciplina. A característica central que conduziu ao resultado satisfatório alcançado foi a coesão entre os integrantes, que se mostraram dedicados e colaborativos.

A seguir, serão apresentadas breves descrições individuais dos membros, com a versão completa disponível na página "Sobre" da aplicação.

- **Ana Paula Moura:** apresenta interesse pelas áreas de jornalismo e administração, ainda indecisa sobre a sua carreira. E a sua experiência profissional inclui o desenvolvimento da Plataforma de Eventos no IFSP CodeLab, em 2022. Na ADA, teve como principal responsabilidade as atividades relacionadas ao backend;
- **Gustavo Santos:** busca uma trajetória na área de segurança da informação e disciplinas correlatas, pretendendo seguir uma carreira acadêmica. E a sua experiência profissional inclui o desenvolvimento da Plataforma de Eventos no IFSP CodeLab, em 2022, e o desempenho na área de mecânica de usinagem, com programação voltada à máquina CNC. Na ADA, atuou como full-stack, com enfoque no desenvolvimento do backend da aplicação e na sua hospedagem;
- **Henrique Baesa:** possui ambições bem definidas para se especializar no campo da inteligência artificial. Como experiência, apresenta certificação pela Universidade de Harvard em Fundamentos da Ciência da Computação, uma medalha na Olimpíada Brasileira de Informática e trabalha como desenvolvedor full-stack. Na ADA, desempenha, principalmente, atividades de backend;
- **Isabella Mazará:** pretende olhar para o abismo da mente humana e espera que ela corresponda o olhar[12]. Logo, seu plano é cursar psicologia e, posteriormente, se especializar em neurociência. E a sua experiência profissional envolve o desenvolvimento da Plataforma de Eventos no IFSP CodeLab, em 2022. Na ADA, atuou como full-stack e seu papel principal foi ser a representante da equipe, concentrando-se em sua organização, na estrutura da ADA, além de tudo que abrange a documentação;
- **Josineudo Arruda:** deseja seguir a carreira da programação, mais precisamente na área de desenvolvimento full-stack e inteligência artificial, além de se interessar por análise de dados. E a sua experiência profissional inclui a participação no IFSP CodeLab e no estágio de Python no Instituto, em 2023. Na ADA, desempenhou um papel full-stack, com enfoque no desenvolvimento de funcionalidades e no front-end, além de ser um dos planejadores das telas;
- **Paulo Muneischi:** possui aspirações na área da programação, principalmente no campo abrangente do desenvolvimento de *softwares* e *websites*, com um grande interesse na área de desenvolvimento de jogos. E a sua experiência profissional

[10] Ada discorre sobre como a Máquina Analítica poderia ser usada para alavancar o progresso da sociedade e não só realizar meros cálculos, além de em sua última nota, escrever um algoritmo para que a máquina pudesse computar a Sequência de Bernoulli. ()

[11] As logos estão no apêndice –

[12] Referência à frase de Friedrich Nietzsche, "E você se olhar durante muito tempo para um abismo, o abismo também olhará para dentro de você.". () Foi um pensador importante à filosofia e serviram de base para Sigmund Freud, criador da psicanálise.

inclui o estágio de Python no Instituto, em 2023. Na ADA, suas responsabilidades incluíram a execução de tarefas full-stack, com um foco na lógica e na programação de processos.

5. Desenvolvimento

Com a finalidade de apresentar uma aplicação em conformidade com os objetivos propostos, é essencial ter em mente as suas funcionalidades, exceções e validações, além da lógica exigida para sua implementação. Ao longo das entregas do projeto, foi possível estabelecer a proposta inicial da equipe, a prova de conceito (vide apêndice B) a entrega parcial (vide apêndice C) e, por fim, a entrega final.

À medida que a ADA é desenvolvida com as tecnologias e ferramentas escolhidas[13], são transpostas as ideias e as lógicas de execução em estruturas para garantir uma compreensão mais ampla. O apêndice – contém um registro abrangente dessas, realizadas pela equipe, incluindo os épicos da aplicação, o modelo de banco de dados, o protótipo, entre outras estruturas relevantes.

Quanto ao idioma escolhido, o principal é o português brasileiro – por ser o nativo do Instituto e da maioria de seus membros – e o secundário é o inglês americano – dada sua universalidade e facilidade de acesso em termos de educação e tradução.

Por fim, a seguir, as funcionalidades/processos serão detalhadamente apresentados, levando em consideração que funções envolvendo o CRUD (presente em todos os aspectos, inclusive nas funcionalidades) e validações menos complexas serão omitidas.[14]

5.1. Funcionalidades

5.1.1. Divisão de usuários

Como mencionado, o sistema de atribuição engloba diversos profissionais, cada um com funções específicas. Para atingir o objetivo proposto, foi primordial implementar três níveis de usuários.

O primeiro é o Administrador, responsável pela organização das configurações gerais do sistema de atribuição. O segundo, o Representante, que realiza alterações limitadas as configurações de seu(s) bloco(s). E o terceiro, o Professor, cujo acesso é direcionado às suas funções particulares, de seleção de preferências e participação da atribuição.

Cada nível de usuário possui uma visualização distinta das telas e existe a flexibilidade de designar um usuário para mais de um nível, permitindo acesso integrado por um único login (prontuário e senha). São visualizações diferentes das telas para cada, além da possibilidade de um usuário ser designado para mais de um papel e acessá-las por meio de um único acesso (prontuário e senha).

Essa abordagem possibilita a centralização dos principais procedimentos, a manutenção das etapas essenciais e a organização mais estruturada do sistema de atribuição.

[13] vide apêndice D

[14] Mais detalhes sobre alguns desses processos e sua representação no banco de dados estão disponíveis no apêndice E, por meio de explicações gráficas.

5.1.2. Cadastro

O cadastro ocorre a partir da adição de usuários por outros usuários, seguindo uma hierarquia de níveis de acesso. O procedimento é organizado da seguinte maneira: os desenvolvedores adicionarão os Administradores, que terão acesso direto à aplicação; por sua vez, os Administradores adicionarão outros Administradores e os Representantes; e, por fim, os Representantes adicionarão os Professores.

Os Representantes e os Professores recém-adicionados receberão um e-mail contendo um link para efetuarem seu cadastro na ADA. Com isso, os seus dados serão integrados aos já armazenados no banco de dados.

Essa abordagem distribuída assegura que a tarefa não fique restrita a um único nível de usuário, garantindo o controle adequado dos usuários e o envio do e-mail a todos, sem deixar um usuário de fora.

5.1.3. Regras

É de suma importância aderir aos regulamentos do IFSP-SPO e do CNE/CEB para manter a legalidade e a integridade da aplicação. Para isso, foram introduzidas algumas regras que abrangem leis e configurações obrigatórias inerentes ao processo.

A regra referente ao Período de Aula (Timeslot) se fundamenta na duração de uma aula. Atualmente, tem a opção de 45 ou 50 minutos. Portanto, foi desenvolvida uma lógica que calcula a diferença entre o início e o fim do horário e a atribui a uma variável, garantindo flexibilidade para futuras mudanças.

A regra referente à Jornada Diária se fundamenta em não exceder o limite de 8 horas diárias de regência de aulas por docente, com a equivalência em quantidade de aulas está na Resolução nº 109/2015, página 9. Além disso, a regra referente à Interjornada se fundamenta na exigência legal de 11 horas de descanso entre duas jornadas de trabalho. Ambas as regras foram validadas na Disponibilidade de Horário do FPA, fornecendo informações corretas para o processo de atribuição e prevenindo uma sobrecarga.

Por outro lado, a regra referente à Quantidade de Estudantes se fundamenta no limite de 25 estudantes por turma e, por conseguinte, um docente por turma. A fim de efetivá-la, uma validação na classe Turma (Classs) foi suficiente.

Por fim, as regras referentes ao Regime de Trabalho se fundamentam na duração do regime do funcionário, que podem ser de 20 ou 40 horas, e, assim, estabelecem um limite de regência de aulas. Nesse caso também foram implementadas diretamente no FPA, com validações para a seleção mínima e máxima de células.

5.1.4. Combos

As combinações de horários, dia e disciplinas, “combos”, são conjuntos de horários sequenciais de um dia, em que está salva a mesma disciplina de uma turma. Essa abordagem resulta na criação de uma classe que engloba dias e horários relacionados a uma ou mais

disciplinas, e esses são registrados durante a montagem da grade horária.

Como mencionado anteriormente, essa funcionalidade desempenha um papel significativo na facilitação do processo de atribuição e, igualmente, no preenchimento do FPA.

5.1.5. FPA

O FPA marca o ponto inicial do processo de atribuição do semestre, uma vez que as configurações necessárias já foram estabelecidas. Seu preenchimento ocorre em duas etapas, a Disponibilidade de Horário e a Preferência de aulas. Na primeira, o docente seleciona o seu regime de trabalho[15], determinando os limites mínimos e máximos de carga horária. Em seguida, seleciona os horários nos quais deseja ou pode trabalhar, respeitando os requisitos mínimos de tempo.

Após a conclusão dessa etapa, o processo segue à segunda etapa, onde seleciona as disciplinas primárias e secundárias desejadas que estão disponíveis nos horários selecionados anteriormente. A ordem de preferência auxilia na coordenação de atribuição, especialmente em situações de conflitos com as preferências de docentes anteriores na fila.

Ao submeter o FPA, as informações são salvas no histórico para referência futura. E, assim, ele desempenha um papel central no processo de atribuição, orientando as escolhas dos docentes e permitindo uma alocação mais eficiente das aulas.

5.1.6. Fila

As filas determinam a sequência em que os docentes terão suas aulas atribuídas, de acordo com o critério estabelecido pelo Administrador. A sua estruturação é realizada pelos Representantes, em cada bloco, e permite alterar a ordem manualmente caso necessário, apesar do critério.

Cabe ressaltar que um docente pode estar inserido em mais de uma fila, e o processo de atribuição distribuirá as aulas para todas simultaneamente. Então, é possibilitada a equidade, garantindo que os docentes recebam atribuições de acordo com suas preferências, sem dependerem de uma única fila.

5.1.7. Atribuição

Se inicia com a automática, que tentará atribuir as aulas primárias do docente. Então, em caso de conflito, tentará atribuir as secundárias, a mesma quantidade que deu conflito nas primárias. E, caso também dê conflito ou não seja possível atingir o mínimo de horas para a regência de aulas, passa à atribuição manual, que enviará um e-mail solicitando que selecione outras disciplinas disponíveis em qualquer horário.

Nas situações em que o docente atua em mais de um bloco, ele pode estar em lugares diferentes da mesma fila. Para isso, o sistema prioriza as disciplinas primárias,

[15] 20 horas, 40 horas, RDE, Temporário ou Substituto

desconsiderando as secundárias, mesmo se forem do primeiro Bloco em que começou a atribuição.

O Celery vai auxiliar no agendamento de tarefas, em que uma função pede para mover o docente para o final da fila, caso não cumpra o prazo, dentro de X horas. Se for uma situação de sucesso, o Celery não executa a função e a tarefa agendada é cancelada àquele docente.

5.1.8. Desativação de docente

A desativação tem como finalidade permitir que um docente não consiga escolher uma disciplina para lecionar caso esteja desativado na mesma. É uma funcionalidade controlada pelo Representante e designada para casos de exceção. Assim, ao selecionar os detalhes do docente, é possível listar a(s) disciplina(s) que o docente será desativado, e ela(s) não ficarão disponíveis na atribuição.

Essa funcionalidade vem da ideia de que a aprendizagem do aluno é proporcionada pela atuação apropriada do docente, tanto no âmbito profissional quanto no pessoal.

5.2. Testes Automatizados

Na execução dos testes unitários e integrados foi empregada a ferramenta Pytest. Eles abarcaram os *apps* User, Timetable, Staff, Common e Attribution, onde foi acordado sobre a prioridade na verificação da execução. Para a realização do procedimento, foram criadas pastas de teste para cada *app*, em virtude do crescimento constante da aplicação. Dentro delas, foram criados arquivos (vide apêndice F, Figura 7) com a implementação de novas funções de teste, a fim de verificar a precisão da execução das funções originais e de confirmar se há o retorno dos resultados esperados. Assim, a ferramenta avaliou a exatidão da saída de dados, como exemplificado no apêndice F, Figura 8 e 9.

Nesse contexto, houve a necessidade de criar classes e funções fictícias (vide apêndice F, Figura 10 e 11) para a preparação dos dados. Vale ressaltar que, embora sejam criadas no banco de dados, as classes são deletadas após a conclusão dos testes, não impactando negativamente no desempenho da aplicação.

Conforme atestado pelo Pytest-cov (Code Coverage / Cobertura de Código), a equipe conseguiu cobrir 60% da ADA (vide apêndice F, Figura 12-14). E, ao longo do processo, foram identificadas falhas (failed), referentes a resultados divergentes em relação ao esperado, e erros (error), referentes a erros nos códigos dos testes. Isso é evidenciado no relatório gerado pela ferramenta, presente no apêndice F, Figura 15-18.

Considerando o exposto, foi possível detectar erros nos códigos das funções da aplicação, falhas que comprometeram o seu correto funcionamento, e algumas funções que se provaram inutilizadas e poderiam ser descartadas, como as que remontam à época da Prova de Conceito (PoC).

6. Considerações Finais

O sistema de atribuição atual do IFSP-SPO claramente evidencia uma necessidade premente de simplificar os procedimentos envolvidos. A proposta da aplicação *web* ADA visa fornecer uma solução por meio de processos abrangentes e robustos, capazes de

enfrentar as complexas demandas do sistema. Isso a partir do reconhecimento da importância da educação eficaz e da necessidade de cumprir os regulamentos legais para atender aos direitos dos usuários.

Ao longo deste artigo, exploramos esses processos e outros aspectos ligados à ADA, desde sua concepção até sua implementação e testes. Cada etapa trouxe consigo lições aprendidas e melhorias, com a aplicação de mudanças contínuas e correções de erros identificados pela equipe e pelos orientadores.

Nesse aspecto, demonstra a maneira como a tecnologia pode beneficiar os usuários sem remover por completo a interferência humana. Os resultados conquistados demonstram a habilidade da equipe de desenvolvimento, que conseguiu criar um sistema autogerenciável e funcional, produzindo bons resultados, mesmo diante dos múltiplos obstáculos.

Através dos resultados obtidos, a ADA serve como ponto de partida para desenvolver uma aplicação que englobe todos os procedimentos do sistema de atribuição atual, descritos no Art. 3 da Portaria n° SPO.071. Consequentemente, no futuro, a aplicação poderá possibilitar a permuta de aulas entre os docentes, seja monitorada ou não pelo Representante. Adicionalmente, poderá incorporar mais leis e exceções, como em situações de substituição e afastamento de docentes, e incorporar o FPA completo, manipulando o processo de designação de atividades de apoio ao ensino e componentes complementares.

A ADA é o primeiro passo para a aplicação da automatização total do sistema de atribuição no IFSP-SPO, na área de DIT, e, ainda, em outros departamentos.

7. Referências

[1] ACIOLI, M. Problemas no sistema de atribuição de aulas prejudicam professores da rede estadual em Mogi. Disponível em: <https://odiariodemogi.net.br/cidades/problemas-no-sistema-de-atribuic-o-de-aulas-prejudicam-professores-da-rede-estadual-em-mogi-1.30131>. Acesso em: 27 ago. 2023.

[2] BATISTA, H. Seção III. Dos períodos de descanso. Disponível em: <https://www.jusbrasil.com.br/doutrina/secao/art-66-secao-iii-dos-periodos-de-descanso-clt-comentada/1198086392#LGL_1943_5-1_A.66>. Acesso em: 27 ago. 2023.

[3] BATISTA, H. Artigo 66 do Decreto Lei nº 5.452 de 01 de Maio de 1943. Disponível em: <https://www.jusbrasil.com.br/topicos/10759018/artigo-66-do-decreto-lei-n-5452-de-01-de-maio-de-1943>. Acesso em: 27 ago. 2023.

[4] CAMPUS SÃO PAULO PIRITUBA. Sobre o IFSP. Disponível em: <https://ptb.ifsp.edu.br/index.php/sobre-o-ifsp>. Acesso em: 27 ago. 2023.

[5] CAMPUS SÃO PAULO. Histórico. Disponível em: <https://spo.ifsp.edu.br/historico>. Acesso em: 27 ago. 2023.

[6] FIGUEIREDO, P.; PAIVA, P. Professores da rede estadual de SP relatam problemas e tumulto no processo de atribuição de aulas. Disponível em: <https://g1.globo.com/sp/sao-paulo/educacao/noticia/

2020/01/24/professores-da-rede-estadual-de-sp-relatam-problemas-e-tumulto-no-processo-de-atribuicao-de-aulas.ghtml>. Acesso em: 27 ago. 2023.

[7] INSTITUTO FEDERAL DE SÃO PAULO. IFSP encerra 2022 com nota máxima em 21 graduações. Disponível em: <https://www.ifsp.edu.br/noticias/3248-cursos-de-graduacao-do-ifsp-sao-avaliados-com-nota-maxima-pelo-mec#:~:text=O%20IFSP%20encerra%202022%20com>. Acesso em: 27 ago. 2023.

[8] MINISTÉRIO DA EDUCAÇÃO. Histórico. Disponível em: <http://portal.mec.gov.br/rede-federal-inicial/historico>. Acesso em: 27 ago. 2023.

[9] MINISTÉRIO DA EDUCAÇÃO. Institucional. Disponível em: <http://portal.mec.gov.br/component/content/article?id=32681:apresentacao>. Acesso em: 27 ago. 2023.

[10] OTRANTO, C. Criação e Implantação dos Institutos Federais de Educação, Ciência e Tecnologia – IFETs. RETTA, Rio de Janeiro, N. 1, p. 89-110, jan-jun, 2010. Disponível em: <https://mapadatese.files.wordpress.com/2013/02/criac3a7c3a3o-e-implantac3a7c3a3o-dos-institutos-federais-cc3a9lia-otranto.pdf>. Acesso em: 27 ago. 2023.

[11] PORTAL DA INDÚSTRIA. Legislação trabalhista: entenda tudo sobre leis trabalhistas (CLT). Disponível em: <https://www.portaldaindustria.com.br/industria-de-a-z/o-que-e-legislacao-trabalhista/>. Acesso em: 27 ago. 2023.

[12] Kaufman, M. (2020). A Brief History of Automation. Disponível em: https://www.exelatech.com/blog/brief-history-automation?language_content_entity=en. Acesso em: 27 ago. 2023.

[13] OVHcloud. O que é automatização?. Disponível em: https://www.ovhcloud.com/pt/learn/what-is-automation/#:~:text=A%20automatiza%C3%A7%C3%A3o%20significa%20a%20substitui%C3%A7%C3%A3o,reduzem%20custos%20e%20o%20stresse. Acesso em: 27 ago. 2023.

[14] OVHcloud (2021). A Brief History of Automation Technology. Disponível em: https://www.imts.com/read/article-details/A-Brief-History-of-Automation-Technology---1/1197/type/Read/1. Acesso em: 27 ago. 2023.

[15] Richardson. D (2021). Top 10 reasons to use automation. Disponível em: https://www.redhat.com/en/blog/top-10-reasons-use-automation. Acesso em: 27 ago. 2023.

[16] FindUp (2023). Como a automação de TI pode ajudar sua empresa?. Disponível em: https://www.ovhcloud.com/pt/learn/what-is-automation/#:~:text=A%20automatiza%C3%A7%C3%A3o%20significa%20a%20substitui%C3%A7%C3%A3o,reduzem%20custos%20e%20o%20stresse. Acesso em: 27 ago. 2023.

[17] Cameron. P. 5 Benefits of Automation: The Advantages of Automation. Disponível em: https://www.fortra.com/resources/guides/automation-advantages-5-benefits-automation. Acesso em: 27 ago. 2023.

[18] Red Hat (2023). O que é a automação de TI?. Disponível em: https://www.redhat.com/pt-br/topics/automation/whats-it-automation. Acesso em: 27 ago. 2023.

[19] CentralIT (2021). VEJA COMO NÃO CONFUNDIR AUTOMAÇÃO E AUTOMATIZAÇÃO. Disponível em: https://centralit.com.br/veja-como-nao-confundir-automacao-e-automatizacao/. Acesso em: 27 ago. 2023.

8. Apêndices

8.1. Apêndice A - Comparação de Aplicações

8.2. Apêndice B - Detalhes sobre a Prova de Conceito

A PoC tem como objetivo verificar a viabilidade do desenvolvimento das funcionalidades propostas inicialmente, através das tecnologias, das ferramentas e do tempo disponíveis. Embora não seja necessária uma elaboração detalhada, sua finalidade é demonstrar a capacidade de aplicar a lógica necessária para a construção da aplicação.

Nessa etapa, a ADA cumpriu suas metas propostas, entregando tanto o cadastro para usuários autorizadas, quanto o principal, a atribuição de aulas. As obrigações solicitadas foram igualmente atendidas: a internacionalização, a hospedagem e a implementação de criptografia.

No entanto, alguns pontos que requerem atenção mais detalhada foram ressaltados pelos orientadores. Tratam-se da necessidade de compreender como os dados são armazenados no banco de dados e a como as classes se relacionam. A criação de um modelo de banco de dados apropriado foi exigida. Além disso, foram identificadas correções necessárias no diagrama de arquitetura e no tratamento de explicações subjetivas relacionadas à seleção das tecnologias e ferramentas utilizadas, entre outros comentários mais específicos.

Em suma, a PoC teve seu objetivo alcançado. A equipe de desenvolvimento da ADA reconheceu a necessidade de fazer alterações nos frameworks utilizados, substituindo o Spring Boot e o Angular pelo Django e Bootstrap, respectivamente, e optou por migrar o sistema de gerenciamento de banco de dados do MySQL para o SQLite3. Quanto aos comentários e observações feitas, foram consideradas e devidamente corrigidas para a entrega subsequente.

8.3. Apêndice C - Detalhes sobre a Entrega Parcial

A Entrega Parcial desempenhou um papel crucial no desenvolvimento das funcionalidades da ADA. Como comprovado na PoC, a possibilidade de desenvolvê-las foi confirmada e, como resultado, foram efetivamente implementadas.

Com a definição oficial da divisão da aplicação em três tipos de usuário, foram entregues diversas funcionalidades: a regra para a determinação dinâmica do período de aula; a grade de aulas com disciplina, horário, período e dias; os combos para sequenciar

aulas da mesma disciplina e turma; o preenchimento digitalizado do FPA, em conformidade com os regulamentes vigentes; a implementação de múltiplas filas de acordo com o critério selecionado e a confirmação do Representante; e a atribuição automática, por meio do software Celery, em conjunto com a atribuição manual, em caso de conflitos de preferências.

Diante dos resultados obtidos, a equipe identificou alterações que necessitavam ser implementadas, são elas: os testes exigidos nas regras da disciplina do projeto; a inclusão de validações específicas; e a padronização, melhoria estética e aplicação de responsividade nas páginas.

Adicionalmente, a equipe constatou que o período de tempo disponível não permitiu a conclusão de algumas tarefas, como a internacionalização de variáveis, a formulação de regras aos períodos de jornada de trabalho e interjornada, a finalização das páginas destinadas aos docentes, e a reorganização de certas URLs.

Durante essa análise dos resultados, também foram identificadas adições benéficas à ADA. Por exemplo, a inclusão de uma aba de notificações, a implementação de uma validação para considerar o período de descanso do docente ao avaliar a possível demora na resposta à atribuição manual, e a busca por melhorias na segurança.

As observações dos orientadores, nesta etapa, foram focadas principalmente em detalhes do front-end e sugestões para melhorar a usabilidade. No entanto, pontos críticos foram destacados, como a necessidade de uma pesquisa mais aprofundada sobre o que implementar para melhorar a segurança, a remoção dos alertas de teste que podem expor dados sensíveis aos usuários, a demanda por personalização das páginas de erro para evitar vulnerabilidades na segurança da informação e nortear o usuário sobre o que ocorreu, e a investigação de um erro na atribuição que causa um loop, quando deveria parar após um número determinado de tentativas. Além disso, foi sugerida a implementação do preenchimento automático das células do FPA ao clicar no período, realizada posteriormente.

Dado o exposto, essa entrega foi essencial ao amplo desenvolvimento da ADA e, igualmente, à identificação de áreas de aprimoramento necessárias para proporcionar uma aplicação mais sofisticada e, consequentemente, uma experiência mais positiva aos usuários.

8.4. Apêndice D - Tecnologias e Ferramentas

Tecnologias

A ADA é uma Single-page Application (SPA) cuja escolha de tecnologias foi fundamentada em um objetivo central: aprendê-las, dominá-las e implementar a aplicação de forma ágil e eficiente para os desenvolvedores. Isso possibilitou direcionar o tempo à lógica das funcionalidades em detrimento de funções mais básicas, como as operações CRUD (Create, Read, Update, Delete).

A linguagem de programação selecionada é o Python, open source e de alto nível, caracterizada por sua forte tipagem dinâmica. Ela se destacou por sua facilidade de compreensão devido à sintaxe concisa e descomplicada, com poucas exigências gramaticais. Para otimizar o desenvolvimento, a escolha recaiu ao *framework web* Django, uma vez que adere ao padrão MTV (Model-Template-View), ao princípio Don't Repeat Yourself

(DRY) e ao nível moderável de opinação. Essa combinação de escolhas já demonstrou sucesso no Sistema Unificado de Administração Pública (SUAP) do IFSP, o que facilita a padronização das aplicações utilizadas pelo Instituto.

Para promover maior interação entre o cliente e o servidor, o AJAX é empregado em conjunto com a linguagem dinâmica JavaScript, a qual permite renderizar as funções e viabiliza interações locais com o conteúdo da página. Esses recursos são obrigatórios ao desenvolver uma SPA e são particularmente imprescindíveis no desenvolvimento de funcionalidades como o FPA.

Outro framework empregado é o Bootstrap, utilizado para a estilização das páginas. Isso simplifica a estilização e garante a responsividade aos desenvolvedores. Para suporte adicional em casos em que não abrange detalhes específicos, recorre-se ao Cascading Style Sheets (CSS).

Quanto ao armazenamento dos dados, optou-se pelo Sistema de Gerenciamento de Banco de Dados (SGBD) SQLite3, por sua capacidade de incorporar o banco de dados diretamente na aplicação (é embutido), eliminando a necessidade de um servidor dedicado, o que está em sintonia com o padrão MTV.

O comportamento da união dessas tecnologias é representada no Diagrama de Arquitetura a seguir:

Por fim, a hospedagem é realizada através da *Amazon Web Services*/Serviços *Web* da Amazon (AWS), para garantir o funcionamento ininterrupto da aplicação, independente de um computador pessoal estar constantemente conectado. Além disso, para aprimorar a identificação, adquiriu-se o domínio https://mottarios.cloud/ por meio da Hostinger, já configurado com o protocolo de segurança HTTPS e com avaliação de nota A pelo Secure Sockets Layer (SSL) Labs.

Ferramentas

No decorrer do desenvolvimento do projeto, foram escolhidas algumas ferramentas com base na familiaridade dos desenvolvedores e em sua conveniência. No que diz respeito ao controle de versões, o TortoiseSVN foi adotado devido à sua organização e segurança, uma vez que já havia sido utilizado em projetos dos anos anteriores na disciplina. Adicionalmente, o GitHub foi escolhido devido à familiaridade e à capacidade de gerenciamento aprimorado dos códigos por meio de branches.

Em relação à documentação, o Overleaf é empregado para manipular os padrões SBC e ABNT com facilidade para os desenvolvedores, possibilitando o compartilhamento e a edição simultânea.

Quanto à programação, a ferramenta escolhida é o Visual Studio Code (VSCode), devido à sua leveza, ao favorecimento da produtividade, por meio do recurso IntelliSense, que a Microsoft recomenda por ajudar "[...] a manter o acompanhamento dos parâmetros que está digitando e a adicionar chamadas a métodos e propriedades pressionando apenas algumas teclas; e comandos para atalhos.". E, novamente, à familiaridade que os membros da equipe já possuem com essa ferramenta.

8.5. Apêndice E - Detalhes Funcionalidades

8.6. Apêndice F - Testes automatizados

Características	Aplicações web		
	ADA	SED	SIG
Direcionada a escolas específicas	✔	✔	✔
Identidade própria da aplicação web	✔	X	X
Log in apenas por pessoas autorizadas	✔	✔	✔
Tutorial sobre o funcionamento da aplicação	X	✔	✔
Configuração dos dados pelos cargos organizadores	✔	✔	?
Redução de procedimentos	✔	X	✔
Estabelecimento e cumprimento de regulamentos e exceções	✔	✔	?
Preenchimento de disponibilidade de horário pelos docentes	✔	✔	✔
Preenchimento de preferência de aulas pelos docentes	✔	✔	✔
Prioridade de escolha	✔	✔	?
Lista para ausência de competência do docente em uma disciplina	✔	X	X
Múltiplas filas de docentes	✔	✔	X
Critério para a fila	✔	✔	X
Várias possibilidades de critérios, com base sólida	✔	X	X
Análise da formação do docente	✔	X	?
Possibilidade de alteração da fila pelos cargos organizadores	✔	✔	X
Atribuição automatizada	✔	✔	X
Execução da atribuição em diferentes períodos, travando as disciplinas	X	✔	X
Atribuição manual pelos docentes	✔	X	X
Atribuição manual pelos cargos organizadores	X	✔	X
Possibilidade de alteração da atribuição final pelos cargos organizadores	X	✔	X
Ativação troca de disciplina entre os docentes, monitorada ou não	✔	X	X
Bot de contato direto para esclarecimento de dúvidas	✔	X	X
Detalhamento sobre o estado atual do docente, caso ausente ou afastado	X	✔	?

Figure 1. Comparação das características apresentadas pelas aplicações na análise de mercado

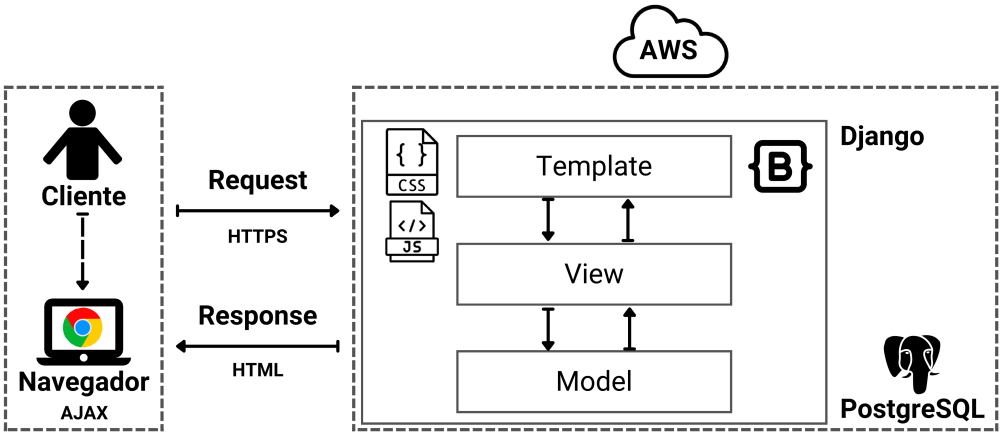


Figure 2. Diagrama de arquitetura da ADA

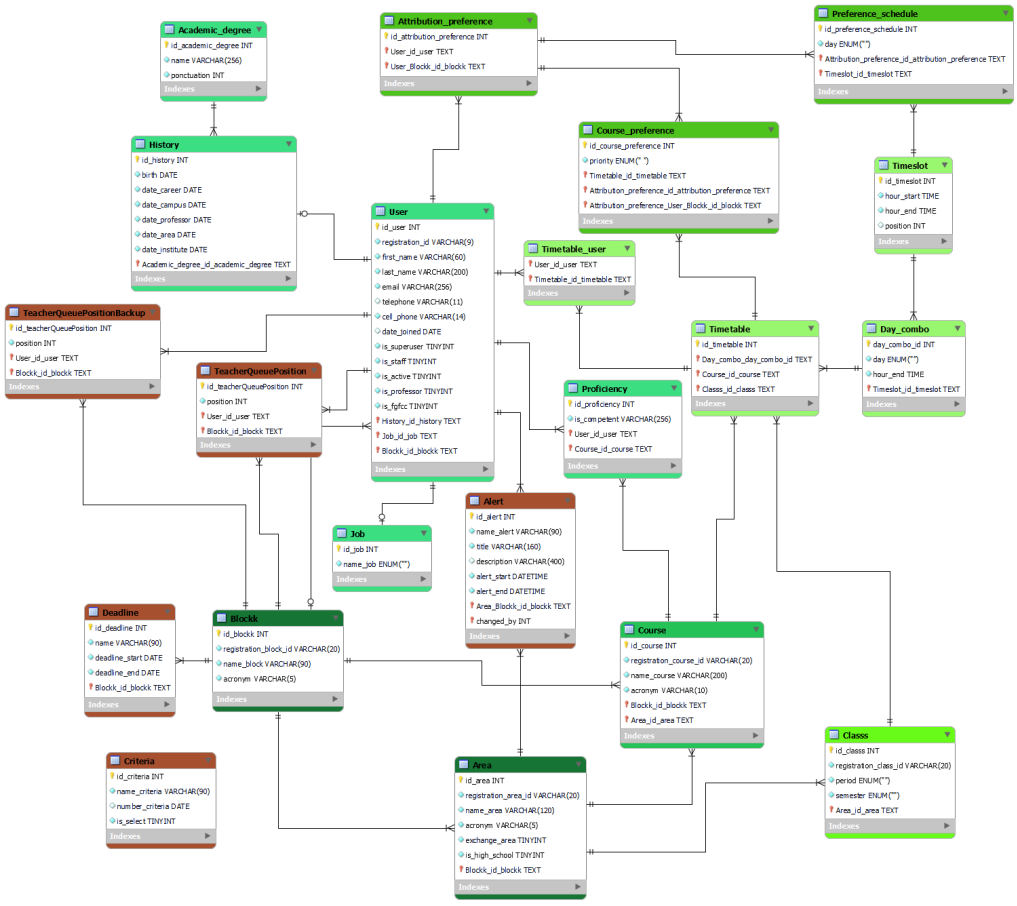


Figure 3. Modelo de banco de dados da ADA

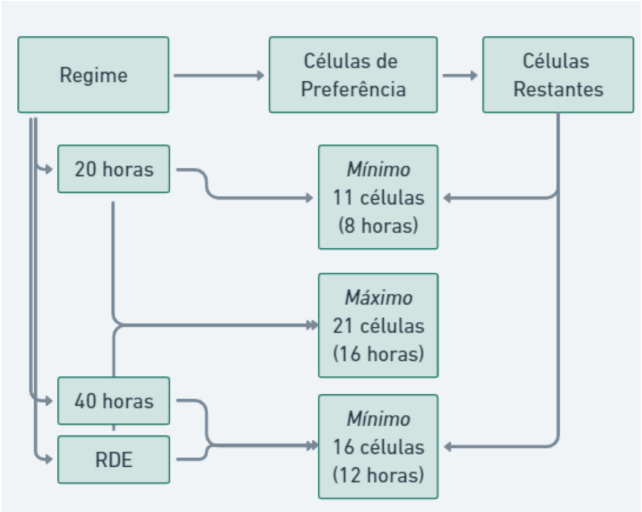


Figure 4. Regras quanto ao regime do docente

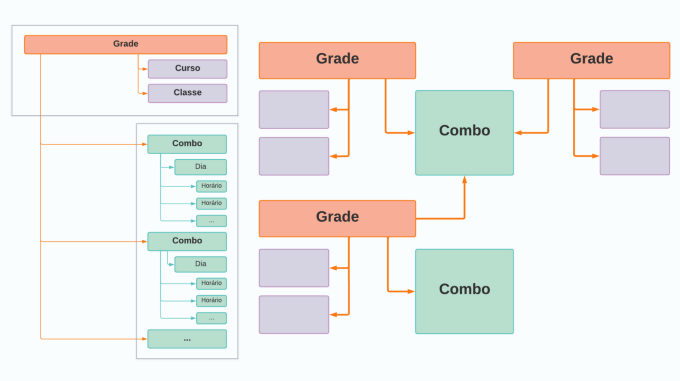


Figure 5. Representação gráfica dos combos

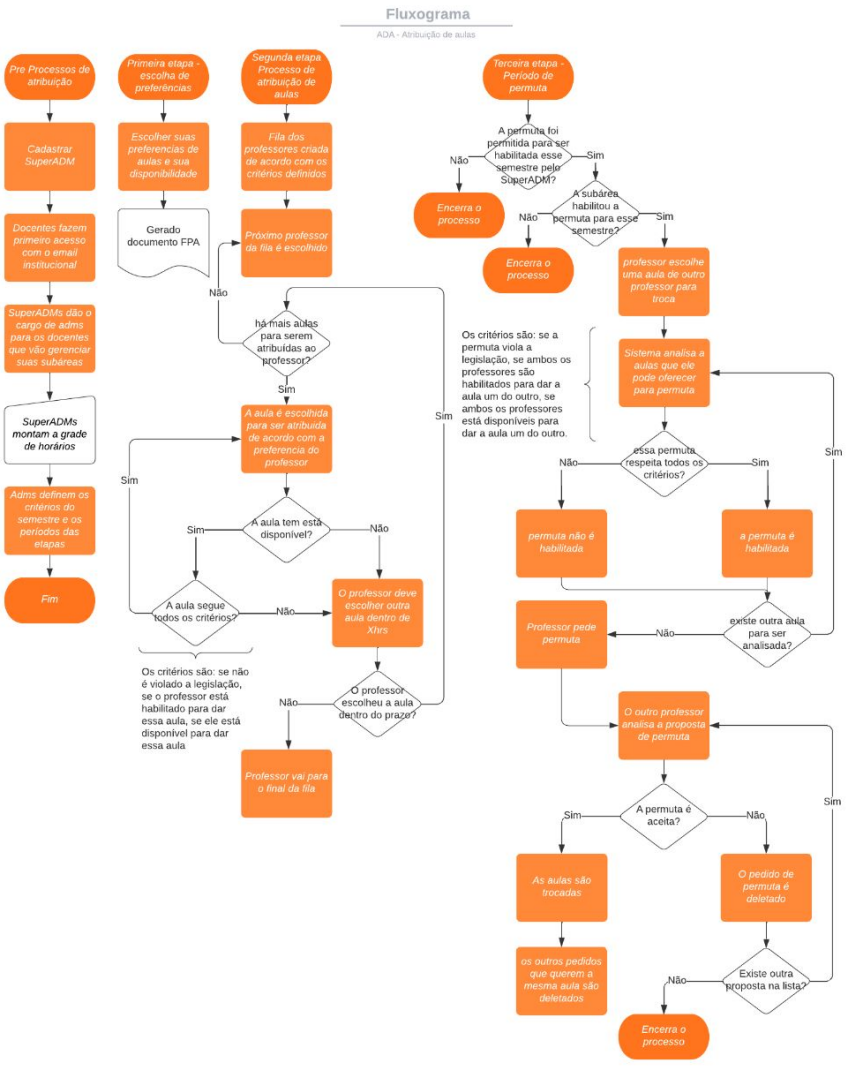


Figure 6. Primeira versão do fluxograma dos processos da ADA, para a PoC

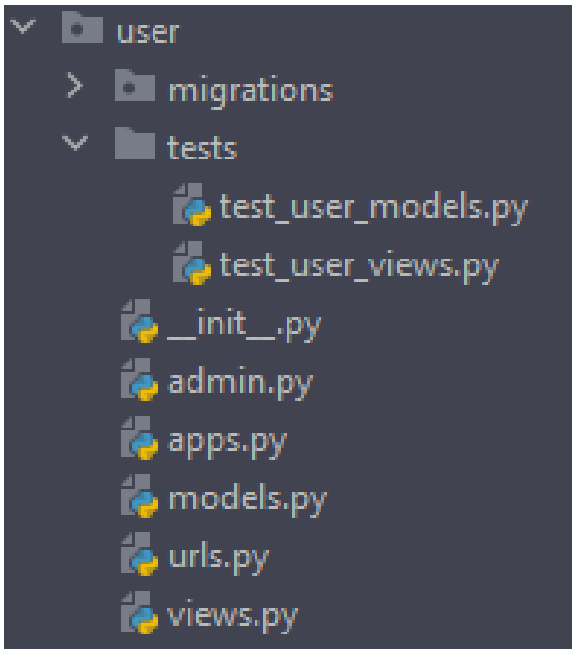


Figure 7. Exemplo das pastas com arquivos de testes

```python
def test_float_to_time_positive():
    # Teste para converter segundos em formato de tempo positivo
    seconds = 3661  # 1 hora, 1 minuto e 1 segundo
    result = float_to_time(seconds)
    assert result.hour == 1  # Deve ser 1 hora
    assert result.minute == 1  # Deve ser 1 minuto
    assert result.second == 1  # Deve ser 1 segundo
```

Figure 8. Exemplo da função teste com a verificação fornecida pelo Pytest

staff\views.py: 51% 314 299 0

```
328 # Erro - TODO
329 @login_required
330 @user_passes_test(is_staff)
331 def class_create(request):
332     if request.method == 'POST':
333         print("funcionou o if")
334         registration_class_id = request.POST.get('registration_class_id')
335         period = request.POST.get('period')
336         semester = request.POST.get('semester')
337         area_id = request.POST.get('area')
338 
339         area = get_object_or_404(Area, id=area_id)
340 
341         classs = Classs.objects.create(registration_class_id=registration_class_id, period=period, semester=semester, area=area)
342         classs.save()
343 
344         return JsonResponse({'message': 'Turma criada com sucesso.'})
345 
346 # Erro - TODO
347 @login_required
348 @user_passes_test(is_staff)
349 def class_delete(request):
350     if request.method == 'POST':
351         print("funcionou o if")
352         print('id')
353         print(request.POST.get('id'))
354         class_id = request.POST.get('id')
355         try:
356             classs = Classs.objects.get(id=class_id)
357             classs.delete()
358             return JsonResponse({'message': 'Turma deletada com sucesso!'})
359         except Course.DoesNotExist:
360             return JsonResponse({'message': 'A turma não existe.'}, status=404)
361 
```

Figure 9. Exemplo de uma verificação que indicou erros no código

```
# Criação de usuários
User.objects.create(registration_id='chico', first_name='Chico', email='chico@email.com', cell_phone='1234567890')
User.objects.create(registration_id='monica', first_name='Monica', email='monicao@email.com',
                    cell_phone='7234567899')
User.objects.create(registration_id='cascão', first_name='Cascão', email='cascao@email.com',
                    cell_phone='9876543210')

# Fazendo requisição à view
response = queue_create(request)

assert response.status_code == 200

assert TeacherQueuePositionBackup.objects.filter(teacher__registration_id='monica', blockk=blockk).exists()
assert TeacherQueuePositionBackup.objects.filter(teacher__registration_id='cascão', blockk=blockk).exists()
assert TeacherQueuePositionBackup.objects.filter(teacher__registration_id='chico', blockk=blockk).exists()

assert TeacherQueuePosition.objects.filter(teacher__registration_id='monica', blockk=blockk).exists()
assert TeacherQueuePosition.objects.filter(teacher__registration_id='cascão', blockk=blockk).exists()
assert TeacherQueuePosition.objects.filter(teacher__registration_id='chico', blockk=blockk).exists()
```

Figure 10. Exemplo de uma classe fictícia

```python
@patch('attribution.views.TeacherQueuePosition.objects')
@patch('attribution.views.validate_timetable')
@patch('attribution.views.assign_timetable_professor')
@patch('attribution.views.professor_to_end_queue')
@pytest.mark.django_db
def test_manual_attribution_save_successful(
        mock_professor_to_end_queue,
        mock_assign_timetable_professor,
        mock_validate_timetable,
        mock_teacher_queue_position_objects,
        timetables,
        professor,
        blockk
):  # Configuração de mock para testar atribuição manual bem-sucedida

    # Configura o mock para retornar o nome do professor corretamente
    mock_teacher_queue_position_objects.get.return_value.teacher.first_name = professor.first_name
    # Configura o mock para retornar a instância do professor correta
    mock_teacher_queue_position_objects.get.return_value.teacher = professor

    # Configura o mock para validar a tabela de horários
    mock_validate_timetable.return_value = True

    # Chama a função de atribuição manual
    result = manual_attribution_save(timetables, professor, blockk)

    # Verifica os resultados esperados
    assert result is None  # O resultado deve ser None para uma atribuição bem-sucedida
    assert mock_teacher_queue_position_objects.filter.called  # Deve ter sido chamado filter do mock
    assert mock_assign_timetable_professor.call_count == len(
        timetables)  # Deve ter sido chamado assign_timetable_professor para cada horário
    assert mock_professor_to_end_queue.called  # Deve ter sido chamado professor_to_end_queue
```

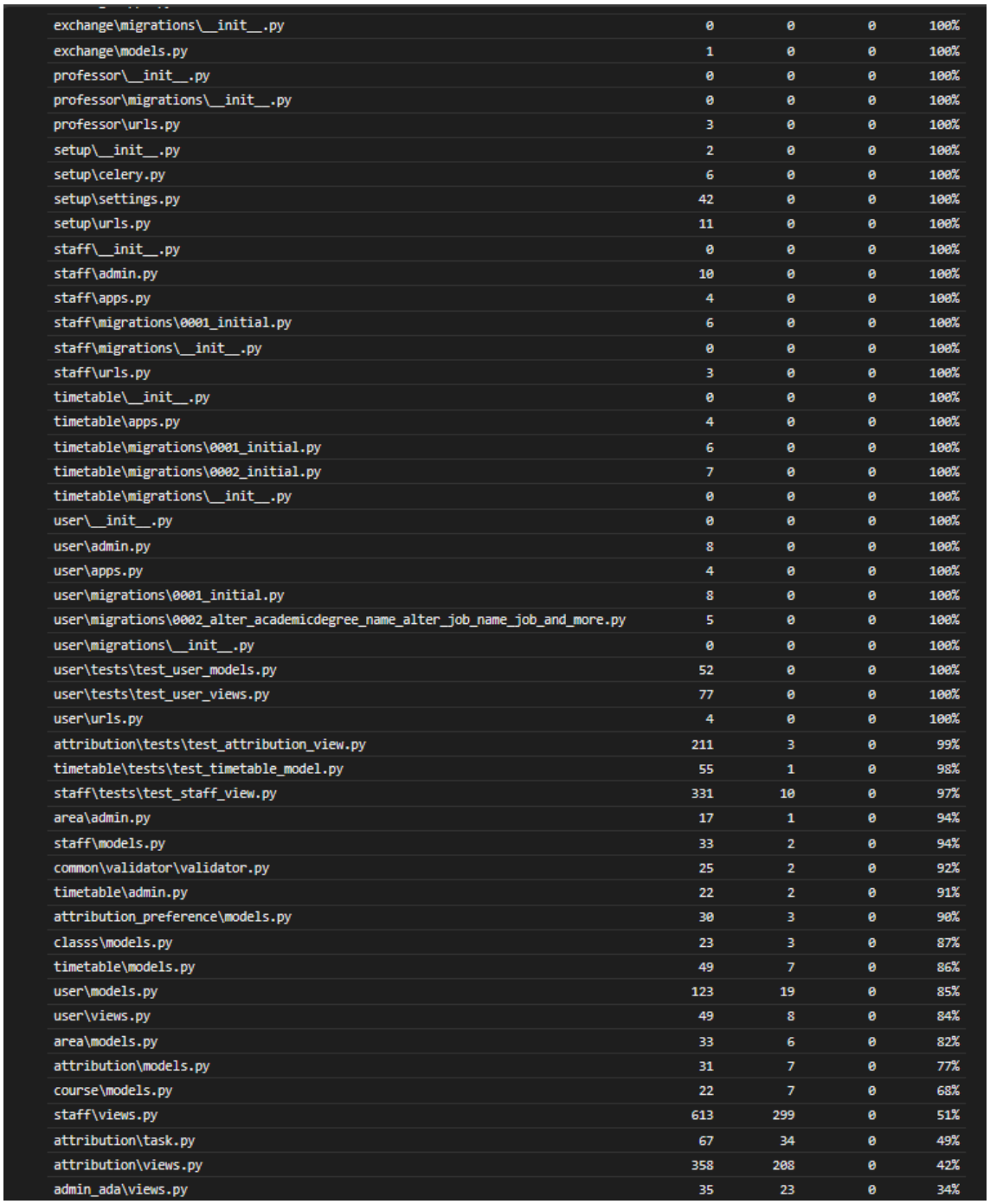
Figure 11. Exemplo de uma função fictícia

Coverage report: 60%

coverage.py v7.3.0, created at 2023-08-28 12:17 -0300

Module	statements	missing	excluded	coverage ↓
admin_ada__init__.py	0	0	0	100%
admin_ada\migrations__init__.py	0	0	0	100%
admin_ada\urls.py	4	0	0	100%
area__init__.py	0	0	0	100%
area\apps.py	4	0	0	100%
area\migrations\0001_initial.py	5	0	0	100%
area\migrations__init__.py	0	0	0	100%
attribution__init__.py	0	0	0	100%
attribution\admin.py	12	0	0	100%
attribution\apps.py	4	0	0	100%
attribution\migrations\0001_initial.py	6	0	0	100%
attribution\migrations\0002_initial.py	7	0	0	100%
attribution\migrations__init__.py	0	0	0	100%
attribution\urls.py	4	0	0	100%
attribution_preference__init__.py	0	0	0	100%
attribution_preference\admin.py	14	0	0	100%
attribution_preference\apps.py	4	0	0	100%
attribution_preference\migrations\0001_initial.py	6	0	0	100%
attribution_preference\migrations\0002_initial.py	6	0	0	100%
attribution_preference\migrations\0003_initial.py	7	0	0	100%
attribution_preference\migrations\0004_course_preference_priority.py	4	0	0	100%
attribution_preference\migrations__init__.py	0	0	0	100%
attribution_preference\urls.py	3	0	0	100%
classs__init__.py	0	0	0	100%
classs\admin.py	6	0	0	100%
classs\apps.py	4	0	0	100%
classs\migrations\0001_initial.py	6	0	0	100%
classs\migrations__init__.py	0	0	0	100%
classs\tests\test_classs_model.py	13	0	0	100%
common__init__.py	0	0	0	100%
common\date_utils__init__.py	3	0	0	100%
common\processors.py	7	0	0	100%
common\tests\test_validator.py	24	0	0	100%
course__init__.py	0	0	0	100%
course\admin.py	6	0	0	100%
course\apps.py	4	0	0	100%
course\migrations\0001_initial.py	6	0	0	100%
course\migrations__init__.py	0	0	0	100%
enums__init__.py	0	0	0	100%
enums\enum.py	15	0	0	100%
exchange__init__.py	0	0	0	100%
exchange\admin.py	1	0	0	100%
exchange\apps.py	4	0	0	100%
exchange\migrations\ init .py	0	0	0	100%

Figure 12. Cobertura dos testes, indicada pela Pytest-cov - Parte 1

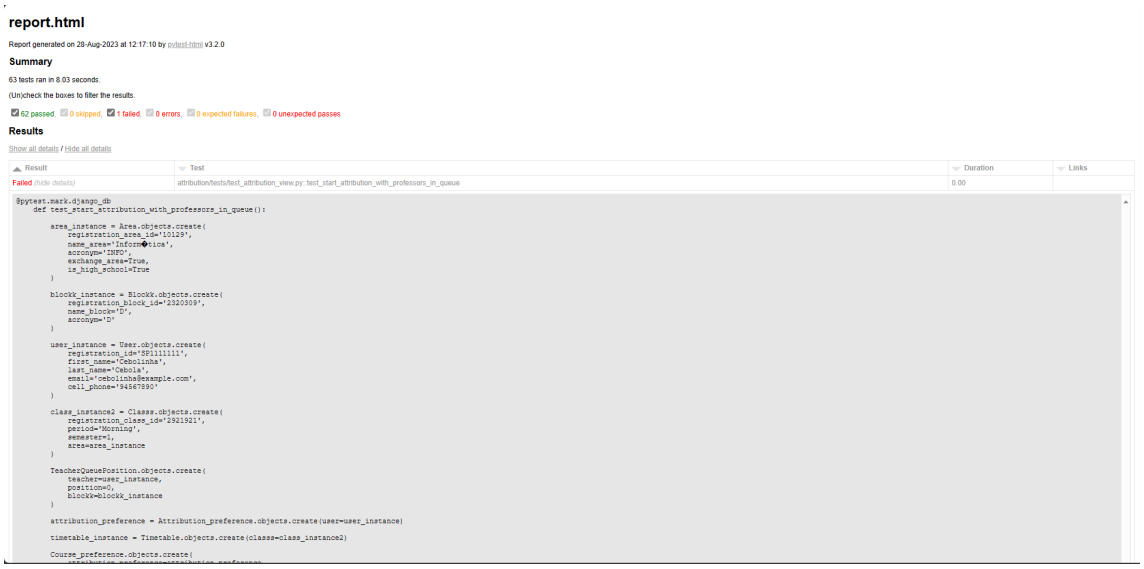


exchange\migrations__init__.py	0	0	0	100%
exchange\models.py	1	0	0	100%
professor__init__.py	0	0	0	100%
professor\migrations__init__.py	0	0	0	100%
professor\urls.py	3	0	0	100%
setup__init__.py	2	0	0	100%
setup\celery.py	6	0	0	100%
setup\settings.py	42	0	0	100%
setup\urls.py	11	0	0	100%
staff__init__.py	0	0	0	100%
staff\admin.py	10	0	0	100%
staff\apps.py	4	0	0	100%
staff\migrations\0001_initial.py	6	0	0	100%
staff\migrations__init__.py	0	0	0	100%
staff\urls.py	3	0	0	100%
timetable__init__.py	0	0	0	100%
timetable\apps.py	4	0	0	100%
timetable\migrations\0001_initial.py	6	0	0	100%
timetable\migrations\0002_initial.py	7	0	0	100%
timetable\migrations__init__.py	0	0	0	100%
user__init__.py	0	0	0	100%
user\admin.py	8	0	0	100%
user\apps.py	4	0	0	100%
user\migrations\0001_initial.py	8	0	0	100%
user\migrations\0002_alter_academicdegree_name_alter_job_name_job_and_more.py	5	0	0	100%
user\migrations__init__.py	0	0	0	100%
user\tests\test_user_models.py	52	0	0	100%
user\tests\test_user_views.py	77	0	0	100%
user\urls.py	4	0	0	100%
attribution\tests\test_attribution_view.py	211	3	0	99%
timetable\tests\test_timetable_model.py	55	1	0	98%
staff\tests\test_staff_view.py	331	10	0	97%
area\admin.py	17	1	0	94%
staff\models.py	33	2	0	94%
common\validator\validator.py	25	2	0	92%
timetable\admin.py	22	2	0	91%
attribution_preference\models.py	30	3	0	90%
classs\models.py	23	3	0	87%
timetable\models.py	49	7	0	86%
user\models.py	123	19	0	85%
user\views.py	49	8	0	84%
area\models.py	33	6	0	82%
attribution\models.py	31	7	0	77%
course\models.py	22	7	0	68%
staff\views.py	613	299	0	51%
attribution\task.py	67	34	0	49%
attribution\views.py	358	208	0	42%
admin_ada\views.py	35	23	0	34%

Figure 13. Cobertura dos testes, indicada pela Pytest-cov - Parte 2

attribution\views.py	358	208	0	42%
admin_ada\views.py	35	23	0	34%
professor\views.py	113	99	0	12%
attribution_preference\views.py	527	501	0	5%
admin_ada\admin.py	1	1	0	0%
admin_ada\apps.py	4	4	0	0%
admin_ada\models.py	1	1	0	0%
admin_ada\tests.py	1	1	0	0%
area\tests.py	1	1	0	0%
area\views.py	1	1	0	0%
attribution_preference\tests.py	1	1	0	0%
classs\tests.py	1	1	0	0%
classs\urls.py	2	2	0	0%
classs\views.py	1	1	0	0%
course\tests.py	1	1	0	0%
course\views.py	1	1	0	0%
exchange\tests.py	1	1	0	0%
exchange\urls.py	4	4	0	0%
exchange\views.py	4	4	0	0%
fixMigrations.py	18	18	0	0%
manage.py	12	12	0	0%
professor\admin.py	1	1	0	0%
professor\apps.py	4	4	0	0%
professor\models.py	1	1	0	0%
professor\tests.py	1	1	0	0%
setup\asgi.py	4	4	0	0%
setup\wsgi.py	4	4	0	0%
timetable\views.py	1	1	0	0%
Total	3290	1316	0	60%

Figure 14. Cobertura dos testes, indicada pela Pytest-cov - Parte 3



report.html

Report generated on 28-Aug-2023 at 12:17:10 by pytest-html v3.2.0

Summary

63 tests ran in 8.03 seconds.

(Un)check the boxes to filter the results.

62 passed, 0 skipped, 1 failed, 0 errors, 0 expected failures, 0 unexpected passes

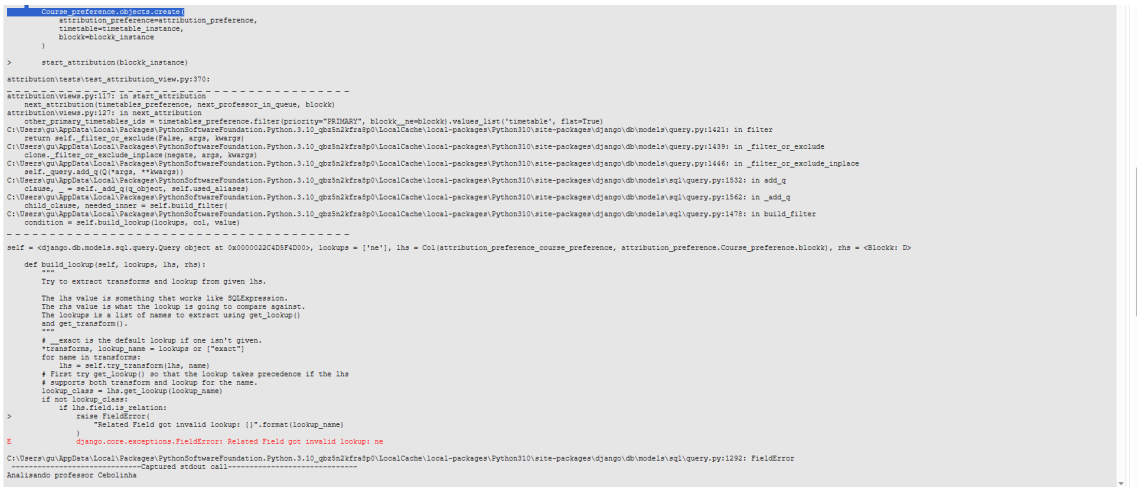
Results

Show all details / Hide all details

Result	Test	Duration	Links
Failed (hide details)	attribution/tests/test_attribution_view.py::test_start_attribution_with_professors_in_queue	0.00	

```
@pytest.mark.django_db
    def test_start_attribution_with_professors_in_queue():

        area_instance = Area.objects.create(
            registration_area_id='10129',
            name_area='Inform\x00tica',
            acronym='INFO',
            exchange_area=True,
            is_high_school=True
        )

        blockk_instance = Blockk.objects.create(
            registration_block_id='2320309',
            name_block='D',
            acronym='D'
        )

        user_instance = User.objects.create(
            registration_id='SP1111111',
            first_name='Cebolinha',
            last_name='Cebola',
            email='cebolinha@example.com',
            cell_phone='94567890'
        )

        class_instance2 = Classs.objects.create(
            registration_class_id='2921921',
            period='Morning',
            semester=1,
            area=area_instance
        )

        TeacherQueuePosition.objects.create(
            teacher=user_instance,
            position=0,
            blockk=blockk_instance
        )

        attribution_preference = Attribution_preference.objects.create(user=user_instance)

        timetable_instance = Timetable.objects.create(classs=class_instance2)

        Course_preference.objects.create(
```

Figure 15. Relatório dos testes - Parte 1



```
            Course_preference.objects.create(
                attribution_preference=attribution_preference,
                timetable=timetable_instance,
                blockk=blockk_instance
            )

>       start_attribution(blockk_instance)

attribution\tests\test_attribution_view.py:370:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
attribution\views.py:117: in start_attribution
    next_attribution(timetables_preference, next_professor_in_queue, blockk)
attribution\views.py:127: in next_attribution
    other_primary_timetables_ids = timetables_preference.filter(priority="PRIMARY", blockk__ne=blockk).values_list('timetable', flat=True)
C:\Users\gu\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\django\db\models\query.py:1421: in filter
    return self._filter_or_exclude(False, args, kwargs)
C:\Users\gu\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\django\db\models\query.py:1439: in _filter_or_exclude
    clone._filter_or_exclude_inplace(negate, args, kwargs)
C:\Users\gu\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\django\db\models\query.py:1446: in _filter_or_exclude_inplace
    self._query.add_q(Q(*args, **kwargs))
C:\Users\gu\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\django\db\models\sql\query.py:1532: in add_q
    clause, _ = self._add_q(q_object, self.used_aliases)
C:\Users\gu\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\django\db\models\sql\query.py:1562: in _add_q
    child_clause, needed_inner = self.build_filter(
C:\Users\gu\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\django\db\models\sql\query.py:1478: in build_filter
    condition = self.build_lookup(lookups, col, value)
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

self = <django.db.models.sql.query.Query object at 0x00000224C4D5F4D00>, lookups = ['ne'], lhs = Col(attribution_preference_course_preference, attribution_preference.Course_preference.blockk), rhs = <Blockk: D>

    def build_lookup(self, lookups, lhs, rhs):
        """
        Try to extract transforms and lookup from given lhs.

        The lhs value is something that works like SQLExpression.
        The rhs value is what the lookup is going to compare against.
        The lookups is a list of names to extract using get_lookup()
        and get_transform().
        """
        # __exact is the default lookup if one isn't given.
        *transforms, lookup_name = lookups or ["exact"]
        for name in transforms:
            lhs = self.try_transform(lhs, name)
        # First try get_lookup() so that the lookup takes precedence if the lhs
        # supports both transform and lookup for the name.
        lookup_class = lhs.get_lookup(lookup_name)
        if not lookup_class:
            if lhs.field.is_relation:
>               raise FieldError(
                    "Related Field got invalid lookup: {}".format(lookup_name)
                )
E               django.core.exceptions.FieldError: Related Field got invalid lookup: ne

C:\Users\gu\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\django\db\models\sql\query.py:1292: FieldError
------------------------------Captured stdout call------------------------------
Analisando professor Cebolinha
```

Figure 16. Relatório dos testes - Parte 2

Passed *(show details)*	attribution/tests/test_attribution_view.py::test_email_test_view	0.83	
Passed *(show details)*	attribution/tests/test_attribution_view.py::test_send_email	0.85	
Passed *(show details)*	attribution/tests/test_attribution_view.py::test_validations_with_no_existing_timetable_user	0.00	
Passed *(show details)*	attribution/tests/test_attribution_view.py::test_validations_with_existing_timetable_user_and_no_assigned_user	0.00	
Passed *(show details)*	attribution/tests/test_attribution_view.py::test_validations_with_existing_timetable_user_and_assigned_user	0.00	
Passed *(show details)*	attribution/tests/test_attribution_view.py::test_manual_attribution_save_successful	0.00	
Passed *(show details)*	attribution/tests/test_attribution_view.py::test_manual_attribution_save_invalid_teacher	0.00	
Passed *(show details)*	attribution/tests/test_attribution_view.py::test_attribution_detail_sucess	0.03	
Passed *(show details)*	attribution/tests/test_attribution_view.py::test_remove_professors_without_preference	0.00	
Passed *(show details)*	attribution/tests/test_attribution_view.py::test_start_attribution_with_empty_queue	0.00	
Passed *(show details)*	attribution/tests/test_attribution_view.py::test_next_attribution_no_primary_timetables	0.00	
Passed *(show details)*	attribution/tests/test_attribution_view.py::test_create_cord	0.00	
Passed *(show details)*	classs/tests/test_classs_model.py::test_classs_semester_high_school	0.00	
Passed *(show details)*	staff/tests/test_staff_view.py::test_attribution_configuration_confirm	0.47	
Passed *(show details)*	staff/tests/test_staff_view.py::test_update_save_with_existing_history	0.21	
Passed *(show details)*	staff/tests/test_staff_view.py::test_update_save_without_existing_history	0.21	
Passed *(show details)*	staff/tests/test_staff_view.py::test_update_save_cleanup	0.21	
Passed *(show details)*	staff/tests/test_staff_view.py::test_update_save_no_access	0.21	
Passed *(show details)*	staff/tests/test_staff_view.py::test_classes_list	0.21	
Passed *(show details)*	staff/tests/test_staff_view.py::test_create_timetable_get_new_timetable_render	0.22	
Passed *(show details)*	staff/tests/test_staff_view.py::test_create_timetable_get_existing_timetable_redirect	0.21	
Passed *(show details)*	staff/tests/test_staff_view.py::test_create_timetable_post_valid_courses	0.21	
Passed *(show details)*	staff/tests/test_staff_view.py::test_create_timetable_post_invalid_class	0.20	
Passed *(show details)*	staff/tests/test_staff_view.py::test_create_timetable_post_invalid_course	0.20	
Passed *(show details)*	staff/tests/test_staff_view.py::test_edit_timetable_get_render	0.22	
Passed *(show details)*	staff/tests/test_staff_view.py::test_create_timetable_invalid_class	0.20	
Passed *(show details)*	staff/tests/test_staff_view.py::test_create_timetable_valid_data	0.21	
Passed *(show details)*	staff/tests/test_staff_view.py::test_edit_timetable_post_valid_data	0.21	
Passed *(show details)*	staff/tests/test_staff_view.py::test_edit_timetable_post_invalid_class	0.20	
Passed *(show details)*	staff/tests/test_staff_view.py::test_edit_timetable_post_invalid_course	0.21	
Passed *(show details)*	staff/tests/test_staff_view.py::test_save_combo_day_existing_day_combo	0.00	
Passed *(show details)*	staff/tests/test_staff_view.py::test_save_combo_day_new_day_combo	0.00	
Passed *(show details)*	staff/tests/test_staff_view.py::test_timetable_combo_saver_multiple_courses	0.01	
Passed *(show details)*	staff/tests/test_staff_view.py::test_timetable_combo_saver_empty_timetable	0.00	
Passed *(show details)*	staff/tests/test_staff_view.py::test_get_selected_field_valid_criteria	0.00	
Passed *(show details)*	staff/tests/test_staff_view.py::test_get_selected_field_invalid_criteria	0.00	
Passed *(show details)*	staff/tests/test_staff_view.py::test_queue_create_success	0.12	

Figure 17. Relatório dos testes - Parte 3

Passed *(show details)*	staff/tests/test_staff_view.py::test_queue_create_success	0.12	
Passed *(show details)*	staff/tests/test_staff_view.py::test_queue_create_failure_blockk_not_found	0.10	
Passed *(show details)*	user/tests/test_user_models.py::test_update_history	0.00	
Passed *(show details)*	user/tests/test_user_models.py::test_update_history_without_academic_degrees	0.00	
Passed *(show details)*	user/tests/test_user_models.py::test_get_first_name_and_last_initial	0.00	
Passed *(show details)*	user/tests/test_user_views.py::test_login_post	0.01	
Passed *(show details)*	user/tests/test_user_views.py::test_home_view_authenticated_staff	0.10	
Passed *(show details)*	user/tests/test_user_views.py::test_home_view_authenticated_staff_only_professor	0.10	
Passed *(show details)*	user/tests/test_user_views.py::test_home_view_authenticated_staff_common_user	0.11	
Passed *(show details)*	user/tests/test_user_views.py::test_handler500	0.00	
Passed *(show details)*	attribution/tests/test_attribution_view.py::test_float_to_time_positive	0.00	
Passed *(show details)*	attribution/tests/test_attribution_view.py::test_float_to_time_zero	0.00	
Passed *(show details)*	attribution/tests/test_attribution_view.py::test_float_to_time_negative	0.00	
Passed *(show details)*	common/tests/test_validator.py::test_validate_uppercase	0.00	
Passed *(show details)*	common/tests/test_validator.py::test_convert_to_uppercase	0.00	
Passed *(show details)*	timetable/tests/test_timetable_model.py::test_validate_incongruity_time_valid	0.00	
Passed *(show details)*	timetable/tests/test_timetable_model.py::test_validate_incongruity_time_invalid_before	0.00	
Passed *(show details)*	timetable/tests/test_timetable_model.py::test_validate_incongruity_time_invalid_same_time	0.00	
Passed *(show details)*	timetable/tests/test_timetable_model.py::test_validate_interrupted_time_no_overlap	0.00	
Passed *(show details)*	timetable/tests/test_timetable_model.py::test_validate_interrupted_time_overlap	0.00	
Passed *(show details)*	timetable/tests/test_timetable_model.py::test_validate_interrupted_time_same_time	0.00	
Passed *(show details)*	timetable/tests/test_timetable_model.py::test_validate_interrupted_time_different_time_and_no_overlap	0.00	
Passed *(show details)*	user/tests/test_user_views.py::test_login_view	0.00	
Passed *(show details)*	user/tests/test_user_views.py::test_signup_view	0.00	
Passed *(show details)*	user/tests/test_user_views.py::test_logout_view	0.00	
Passed *(show details)*	user/tests/test_user_views.py::test_404_handler	0.02	

Figure 18. Relatório dos testes - Parte 4