Instituto de Matemática e Estatística
Universidade de São Paulo

Trabalho de Formatura Supervisionado

MatrUSP

Autores: Bruno de Oliveira Endo, José Ernesto Young Rodrigues e Rafael Marinaro Verona

Supervisores: José Coelho de Pina e Pedro Paulo Vezzá Campos

Agradecimentos

Aos nossos orientadores, José Coelho de Pina Junior e Pedro Paulo Vezzá Campos, sem o seu auxílio a realização deste trabalho não seria possível.

As nossas famílias por terem nos apoiado durante todo esse periodo.

A todos os nossos colegas que nos acompanharam durante a nossa graduação, em especial ao grande amigo Gustavo Spelzon Caparica, pela grande ajuda em momentos importantes da graduação e da vida, você é um grande companheiro em nossas vidas.

Resumo

O planejamento e organização das atividades a serem realizadas, em um próximo semestre, pelos discentes é uma parte importante na vida acadêmica e pessoal dos estudantes. O sistema JupiterWeb não supre essa necessidade, uma vez que as informações sobre disciplinas a serem oferecidas estão dispersas, não exibe graficamente as possíveis grades horárias antes do período de matrícula e não calcula as diversas possibilidades de grades de acordo com a variação das turmas para um determinado conjunto de disciplinas.

O MatrUSP surgiu para resolver esses problemas e melhorar a experiência do usuário na matrícula e no seu gerenciamento de horários pessoais. Através dele é possível verificar o horário de todas as matérias disponibilizadas pela Universidade de São Paulo, verificar conflitos de horários de uma forma clara e, ainda, verificar todas as possíveis combinações de uma determinada seleção de matérias para um rápido planejamento de rotina.

Os principais objetivos deste projeto são modernizar e incrementar o sistema MatrUSP, utilizando tecnologías web a fim de melhorar a usabilidade, reformular e adicionar ferramentas que facilitem ainda mais o gerenciamento do tempo, aproveitando-se também de plataformas externas amplamente utilizadas como Google Calendar e Facebook para visualizar e compartilhar esse tipo de informação.

Sumário

1 Introdução

1.1 O que é o Matrusp

O MatrUSP é um sistema para auxiliar os alunos da USP a planejarem sua matrícula. Dentro do sistema, o usuário pode escolher as disciplinas, e respectivas turmas, que pretende cursar no próximo semestre, e colocá-las em uma ordem de prioridade. Então, o software calcula todas as possibilidades de combinações das turmas selecionadas para que todos os eventos não conflitem e exibe graficamente ao usuário, permitindo que ele escolha, de forma prática, a combinação que mais lhe agrada.

1.2 Motivação

O sistema da universidade, JúpiterWeb[1], provê a funcionalidade de seleção de disciplinas e realização da matrícula, porém não fornece ao aluno uma forma prática de visualizar sua grade horária antes de confirmar a matrícula ou sequer qualquer tipo de visualização antes do período de matrícula, o que dificulta seu planejamento e realização, pois só é possível verificar conflitos entre as disciplinas ao se tentar salvar uma grade. O MatrUSP surgiu para melhorar essa experiência além de trazer uma série de outras funcionalidades que serão descritas mais adiante.

Além disso, durante a graduação, mesmo criando uma quantidade grande de softwares na forma de exercícios programas, não tivemos a oportunidade acadêmica de criar algo que terá grande impacto na vida de outras pessoas. Desenvolver o MatrUSP foi uma oportunidade única para deixarmos como legado um produto que atinge e melhora a vida de grande parte dos alunos da comunidade uspiana.

O que torna o desenvolvimento deste software ainda mais recompensador é o fato de sermos, nós três, usuários ativos do sistema. Desde quando começamos a incluir disciplinas eletivas nas nossas grades semestrais, utilizar o MatrUSP foi muito gratificante por ele já resolver diversos problemas existentes no fluxo de matrícula. Ainda assim, sempre nos pareceu que havia espaço para aumentar o número de funcionalidades, incluir ferramentas que se mostravam úteis para nossos colegas mais próximos e para nós mesmos, e melhorar como tudo era exibido ao usuário, melhorando a interface gráfica.

1.3 Proposta inicial de trablaho

Nossa proposta de trabalho é criar uma nova versão do MatrUSP que seja melhor em diversas frentes. Pretendemos reformular o design da página melhorando a interação dos usuários, nos apoiando em conceitos de interação humano-computador (IHC). Também pretendemos refatorar o código, melhorando consideravelmente a sua manutenção. Tornar o sistema flexível a ponto de ser acessível por outras universidades é mais um dos objetivos, tentando diminuir ao máximo o número de adaptações necessárias para atender às particularidades de cada universidade.

Além disso, buscamos construir versões específicas do MatrUSP para plataformas móveis (Android, iOS e Windows Phone) caso haja tempo, pretendemos pesquisar e implementar uma forma de recomendação de disciplinas aos usuários. Existem muitos cenários onde o aluno tem pouca liberdade para escolher o que cursar no semestre e, com isso em mente, achamos interessante tentar pré-popular a grade horária com as disciplinas obrigatórias do semestre.

[1] https://uspdigital.usp.br/jupiterweb/

2 O sistema MatrUSP

2.1 História

As matérias optativas, na USP, proporcionam grande flexibilidade na grade curricular dos alunos, porém, juntamente com as matérias cursadas fora de seu período ideal, surge a dificuldade de realizar o planejamento antecipado desses horários, quando, além destas, são contempladas disciplinas obrigatórias e atividades extracurriculares dos discentes.

Isto acontece, pois a USP, e em particular o sistema JúpiterWeb, não possui uma ferramenta capaz de agrupar esses dados e produzir uma resposta gráfica que compute as diversas possibilidades de grades horárias, de acordo com as disciplinas escolhidas. Além disso, ele não divulga de forma clara as disciplinas a serem oferecidas no semestre seguinte antes de abrir o período de matrícula, ainda que estas informações já estejam disponíveis. Assim, estes dois pontos tornam o planejamento sobre quais matérias cursar em um determinado semestre uma atividade potencialmente complexa de ser realizada.

O GRAMA (GRAde de MAtrícula) surgiu para tentar solucionar esse problema, que também existia na Universidade Federal de Santa Catarina, a UFSC, onde foi criado por um aluno de Engenharia de Produção. Esse sistema possuía apoio da UFSC, mas o perdeu e foi descontinuado quando o seu criador tentou usá-lo para promoção pessoal. Para substituí-lo, um programa similar foi criado, o MatrUFSC, que posteriormente adotou o nome CAPIM (Combinador Automático de Possibilidades Interativo de Matrícula).

Em 2012, o estudante de graduação no Bacharelado em Ciência da Computação (BCC) do Instituto de Matemática e Estatística (IME) da USP, Pedro Paulo Vezzá Campos, que havia estudado na UFSC, vivenciando os problemas descritos anteriormente e tendo contato com a solução oferecida pelo CAPIM, resolveu criar uma ferramenta semelhante para a nossa faculdade, o MatrUSP, importando parte do código do sistema MatrUFSC e adaptando-o conforme as necessidades específicas da comunidade uspiana.

2.2 O porquê

O MatrUSP é um sistema que tem como objetivo auxiliar os alunos da USP a planejarem suas matrículas, isto é, quais disciplinas escolher para cursar em cada semestre letivo. Ele nasceu da demanda por um sistema que desse suporte ao processo de matrícula, nos pontos em que o sistema da universidade deixa a desejar.

O JúpiterWeb tem a funcionalidade de seleção de matérias e efetivação da matrícula, porém peca em alguns quesitos. No momento de seleção de disciplinas, não se sabe se cada nova inclusão é compatível com as demais disciplinas já incluídas, isto é, se há uma combinação de turmas de cada disciplina que permita ao aluno cursar todas aquelas que ele selecionou, uma vez que só é percebido o conflito de horário de matérias após a tentativa de realização da matrícula.

Além disso, mesmo quando o conflito de disciplinas não se apresenta como problema (apesar de ser a principal fonte deles), há ainda dois outros grandes complicadores: a busca por diferentes combinações de horários e a visualização da grade horária. Quando o aluno opta, por exemplo, em fazer duas ou mais disciplinas que possuam mais de uma turma com horários compatíveis, surge a possibilidade de optar por diferentes grades horárias. Apenas com o JúpiterWeb, a única forma de encontrar combinações de turmas compatíveis, para então escolher a melhor, é procurando uma a uma: olhando o horário de cada turma e verificando se uma determinada combinação é possível.

Outro problema que o MatrUSP busca sanar é da visualização da grade horária. No sistema oficial da universidade, quando a matrícula é realizada, surge a possibilidade de observar sua grade horária, entretanto esta não se apresenta de uma forma intuitiva. Como apresentado na figura 1, se uma disciplina foi selecionada com horário de terças-feiras das 10:00 às 11:40 (na figura, MAC0300) e outra de segundas-feiras das 09:00 às 11:40 (CMU0449), o que se observa na tabela de grade horária é que duas linhas são criadas e, visualmente, MAC0300 aparece abaixo do fim de CMU0449. Desta forma, a impressão que se tem, em um primeiro momento, é de que as disciplinas ocorrem em horários disjuntos (desconsiderando-se que são dias diferentes), enquanto que elas ocorrem praticamente no mesmo horário.

Figura 1: Três disciplinas que possuem horários sobrepostos apresentadas no Júpiter Web, parecem ter horários que não se sobrepõe.

2.3 MatrUSP na matrícula

O MatrUSP, portanto, é uma ferramenta para ser utilizada em dois momentos. Primeiramente, auxilia os alunos a encontrarem um conjunto de disciplinas e turmas que lhes seja mais agradável ou compatível, logo antes da realização da matrícula, que será realizada com base nas decisões que esse sistema permitiu.

Ademais, como foi dito acima, ele soluciona o problema de visualizar a grade horária de forma intuitiva e pode ser utilizado para registrar, graficamente, como as turmas selecionadas pelo usuário na matrícula se distribuirão pelas semanas do semestre em questão.

3 Metas do trabalho

3.1 Motivação

Desde as primeiras reuniões concordamos em alguns objetivos bastante claros: ao final do trabalho gostaríamos de ter um software que proporcionasse uma experiência de usuário melhor do que a existente no momento, que entregasse aos usuários novas funcionalidades e que a manutenção da base de código se tornasse uma tarefa muito mais fácil e agradável.

Enxergamos que havia muito espaço para desenvolver nestas três frentes, uma vez que, rapidamente, conseguimos pensar em várias funcionalidades interessantes; o design gráfico estava bastante distante do que é visto, hoje em dia, em websites mais modernos como, por exemplo, o Facebook[2] e o Google Drive[3]; e tentar compreender todo o código, naquele momento, era muito difícil.

Com isso, queríamos realizar um trabalho para desenvolver ainda mais uma ferramenta que já ajudava muitos alunos, para ajudá-los de uma forma ainda melhor. Queríamos torná-la mais fácil e agradável de utilizar, com mais recursos para melhorar sua utilização e diminuindo muito a barreira de entrada para qualquer próximo desenvolvedor que queira se envolver no projeto.

Portanto, começamos o planejamento do trabalho observando duas frentes distintas: como se daria o desenvolvimento, isto é, como escreveríamos o código e por onde começaríamos, e onde desejávamos chegar, ou seja, quais as funcionalidades e design desejados e em qual ponto gostaríamos de entregar o software no fim do ano.

3.2 Código

No início do trabalho a opção mais clara e direta parecia ser trabalharmos em cima do código legado e modificarmos ele para atender às nossas novas expectativas. Entretanto, após vários dias de contato com o código, então atual, do MatrUSP, percebemos que teríamos um trabalho enorme para compreendê-lo, um trabalho ainda maior para melhorá-lo e incluir as novidades que pretendíamos e, como se isso não bastasse para mudarmos de ideia, grande parte desta dificuldade seria eventualmente revivida, caso algum desenvolvedor viesse a se interessar por trabalhar no software.

Com todas estas questões discutidas entre nós e nossos orientadores, tomamos a decisão de reescrever todo o código. Fizemos isso com o objetivo de construirmos um ambiente onde toda linha de código escrita fosse bastante clara quanto ao objetivo a que ela se propunha. Onde isso não era possível, comentaríamos o código para facilitar sua compreensão. Além disso, nos propusemos a criar uma documentação que abrangesse o software todo. Desta forma, para um desenvolvedor criar intimidade com as engrenagens do MatrUSP, ele não teria muitas dificuldades de fazê-lo apenas lendo o código e a documentação disponível.

Pensando na proposta de tornar o sistema acessível por outras universidades, tanto nacionais quanto internacionais, também foi colocado como meta tornar essa nova base de código internacional. Para isso, todos os termos utilizados exclusivamente dentro do sistema e da documentação, como nomes de variáveis ou descrições de classes, seriam grafados em inglês, facilitando a transição para outros desenvolvedores de qualquer parte do mundo.

3.3 Funcionalidades

Para começar a cogitar quais funcionalidades iríamos incluir na nova versão do MatrUSP, pesquisamos como é realizada a matrícula em diversas faculdades e universidades. Entretanto, ficou claro que a forma como isso é feito não é uma preocupação na maioria das instituições pesquisadas, nacionais e internacionais, e todas elas apresentavam problemas muito parecidos, na realização dessa tarefa, com os que o nosso sistema se propunha a resolver.

O único sistema que encontramos que resolve os problemas aos quais o MatrUSP se propõe a resolver foi o sistema da UNICAMP: GDE[4]. Ele é um sistema muito maior do que o MatrUSP, é uma rede social contendo informações do restaurante universitário, mapa do campus, ranking dos professores, entre outros.

[2] https://www.facebook.com/
[3] https://drive.google.com
[4] grade.daconline.unicamp.br/visoes/Bemvindo.php

A criação da grade horária é apenas um dos serviços realizados por ele, o qual não continha boas ideias de funcionalidades desconexas do ambiente da rede social. Desta forma, não tiramos proveito criativo deste sistema.

Sendo assim, prosseguimos decidindo arbitrariamente o que adicionar às funcionalidades que já existiam na versão antiga, mas sempre tentando colocar em primeiro lugar e imaginar os interessantes dos usuários, visto que nós três somos também usuários do sistema, e chegamos às seguintes funcionalidades:

- Fazer download de um PDF com a grade horária
- Compartilhamento de grade horária por e-mail
- Compartilhamento de grade horária por Facebook
- Adicionar grade horária ao calendário do Google
- Permitir autenticação de usuários (e.g. login pelo Facebook)
- Salvar histórico de um usuário por semestre
- Recomendar disciplinas de interesse do usuário
- Permitir a restrição de busca de disciplinas por instituto
- Garantir o funcionamento da página em aparelhos móveis construindo versões específicas para as principais plataformas
- Isolar a parte de extração de informações do JúpiterWeb e construir uma API
- Permitir que outras universidades utilizem o sistema de forma prática e fácil

Além destas, quase todas as funcionalidades antigas também foram colocadas como requisitos, são elas:

- Criação de uma grade horária pela inclusão e exclusão de disciplinas
- Cálculo das diferentes combinações possíveis
- Mudança de prioridade de uma disciplina no cálculo das combinações
- Salvar o estado no LocalStorage do navegador
- Salvar o estado por meio de um identificador no servidor
- Permitir diferentes planos de grades horárias
- Inclusão de atividades extracurriculares personalizadas

Com tais objetivos colocados em perspectiva, tínhamos a intenção de permitir que o usuário pudesse realizar ainda mais tarefas que fossem úteis no momento de planejamento de matrícula e também após sua definição.

3.4 Design

Certas questões sobre como o MatrUSP deveria se transformar já se mostraram consensuais desde o primeiro momento que nos reunimos. Todos nós concordamos, e colocamos como um dos objetivos iniciais, refazer o design da página do MatrUSP.

O design antigo era algo característico de um desenho que apenas permitia expor as funcionalidades que existiam no sistema. Entretanto, acreditávamos que era possível mais do que isso e planejamos redesenhar a parte gráfica de modo a torná-la mais bonita, mais intuitiva e mais fácil de usar. Parte das funcionalidades eram pouco notadas pela pobre interface gráfica e queríamos torná-las mais evidentes, alcançando aqueles usuários que, simplesmente, não as haviam notado.

Tornar o MatrUSP mais bonito, também era algo importante para manter o software moderno e atual na perspectiva dos usuários. Uma página com aparência nova e mais próxima aos exemplos mais modernos da indústria, geraria uma sensação de atenção, por parte dos desenvolvedores (nós), a todo o software na percepção dos usuários, potencialmente estabelecendo um maior nível de confiança, fidelidade e recomendações.

4 Decisões de desenvolvimento

4.1 Metodologia

Quando cursamos a matéria de Engenharia de Software temos a impressão de que os tópicos tratados são de pouca relevância em um projeto de verdade, contudo na hora de construir um software do zero percebemos a real importância dessa área. O resultado final do código do MatrUSP seria muito inferior se não tivéssemos pesquisado e utilizado alguns métodos e práticas de como construir um sistema, desde o planejamento para a reutilização de código até a sua estruturação.

Por não termos experiência no desenvolvimento de softwares da proporção do MatrUSP, decidimos que antes de ser tomada qualquer decisão seria necessário pesquisar outros softwares e técnicas para decidirmos como iríamos estruturá-lo. Graças a tais pesquisas conseguimos tomar decisões importantes para o desenvolvimento do sistema e avaliar o que seria ou não possível realizar com o prazo que teríamos.

Adotamos esta metodologia ao longo de todo o desenvolvimento do projeto. Antes de tomar uma decisão, seja de estruturação do código ou de design gráfico, pesquisamos diferentes alternativas e opiniões de pessoas experientes na área para, então, em reunião decidirmos qual caminho seguir, quais objetivos solidificar e, finalmente, começar o desenvolvimento do item abordado.

4.2 Tecnologias

4.2.1 HTML

HyperText Markup Language é uma linguagem de marcação utilizada na construção de páginas Web[5]. Ela pode ser gerada estaticamente ou dinamicamente, isto é, podemos prover um código fixo ou construí-lo de acordo com a interação particular de cada usuário. No nosso caso, utilizamos uma estratégia híbrida: uma parte do MatrUSP é comum a qualquer usuário e, portanto faz parte de um arquivo HTML estático; já o que, na página, diz respeito às disciplinas escolhidas varia e, sendo assim, tem seu código HTML gerado dinamicamente.

4.2.2 JavaScript

É uma das linguagens mais utilizadas para o desenvolvimento Web, por ser a linguagem interpretada pelos navegadores web, é a partir de diretivas em JavaScript que torna-se possível a interação com maior dinamismo entre usuário e aplicação sem a constante interferência de requisições ao servidor. Trabalhar com ela facilita o desenvolvimento por existir uma comunidade grande e ativa, permitindo uma fluidez no que diz respeito à solução de dúvidas e conselhos sobre boas formas de realizar alguma tarefa. Além disso, o código do MatrUSP antigo foi escrito em JavaScript e, ainda que o tenhamos reescrito, foi muito recompensador poder comparar os métodos de implementação de algumas funcionalidades.

4.2.3 SASS/CSS

Cascading Style Sheets, ou CSS, é uma linguagem que permite definir a apresentação visual de um documento de linguagem de marcação como o HTML[6]. Com ele, torna-se mais fácil a separação entre o conteúdo e a forma do documento final. Entretanto, o desenvolvimento em CSS é notavelmente burocrático por exigir que cada estilo seja aplicado explicitamente a cada elemento alvo. Ainda, dependendo do parâmetro sendo definido, a regra pode ou não ser aplicada a todos elementos aninhados naquele elemento alvo, propagando em cascata (cascading).

Por causa disso, é comum que mais de uma definição seja aplicada ao mesmo parâmetro de um mesmo elemento e, então, faz-se necessário escolher qual destas definições será, de fato, aplicada. Para resolver este problema, existe uma regra hierárquica implícita, pela qual certas definições sobrescrevem outras. Esta regra leva em consideração o nível de especificidade do identificador do objeto ao qual pretende-se aplicar um estilo e a ordem em que tais identificadores aparecem no código.

[5] https://pt.wikipedia.org/wiki/HTML
[6] https://pt.wikipedia.org/wiki/Cascading_Style_Sheets

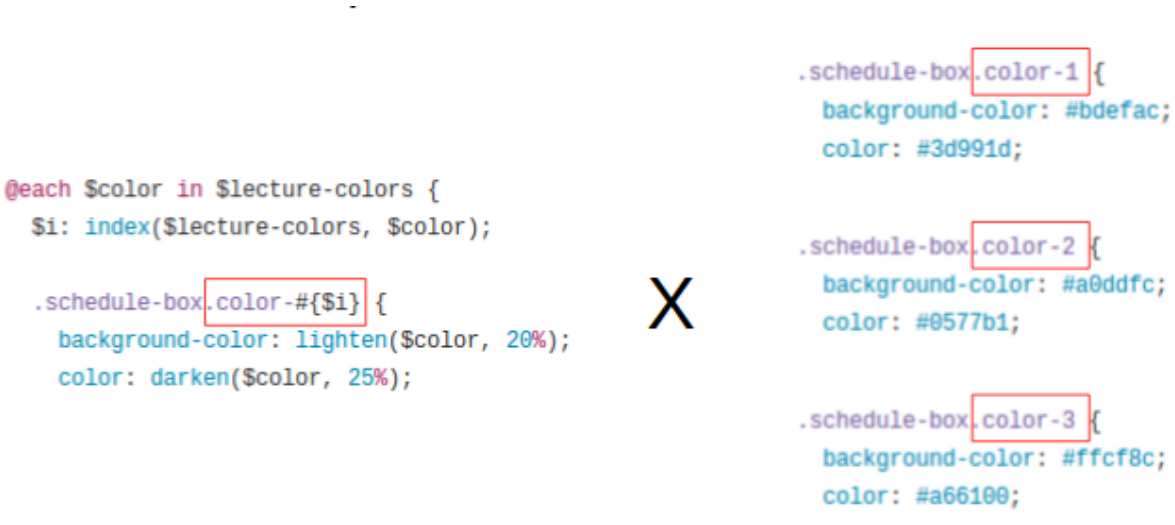

Figura 2: Comparação SASS e CSS

Para tornar o código mais compreensível aos desenvolvedores, optamos por utilizar o SASS, que é uma ferramenta que auxilia no desenvolvimento do CSS. Ele é um pré processador que recebe arquivos .sass e gera arquivos .css como resultado. Ele permite a criação de loops, variáveis, e a criação de escopo entre identificadores, melhorando a leitura do código, facilitando a criação de estilos e tornando factível a utilização de temas. Além disso, por apenas pré-processar o texto, não adiciona nenhum overhead no lado do cliente, isto é, nenhuma biblioteca adicional precisa ser transferida para o browser do usuário.

4.2.4 Git

O controle de versão de softwares permite que tenhamos um histórico de "versões" do projeto que, além de mostrar as mudanças que foram feitas no código com o tempo, permite, caso haja necessidade, voltar para versões passadas e reverter mudanças facilmente. Essa habilidade, de permitir a comparação de duas versões ou reverter mudanças, torna-o imprescindível em projetos que não são realizados por apenas uma pessoa.

O git é um sistema de controle de versão distribuído, ou seja, cada desenvolvedor tem uma cópia de todo histórico das versões, de todo o sistema. Isso torna esse tipo de ambiente muito mais seguro contra falhas, pois mesmo que o servidor perca as informações, cada um dos desenvolvedores tem uma cópia de segurança na sua máquina.

4.3 Dependências de Terceiros

4.3.1 API Facebook

A API do Facebook foi utilizada para duas funcionalidades:

- Exibir o botão "Like" no rodapé da página do novo MatrUSP, que ao ser pressionado e liberado gera um evento "Like", gerenciado pelo Facebook, entre a página MatrUSP no Facebook (www.facebook.com/MatrUSP) e o usuário do Facebook logado no momento da ação no browser em uso. Essa funcionalidade já pertencia à versão anterior do MatrUSP e foi preservada.

- Permitir o compartilhamento de uma grade horária configurada no MatrUSP no Facebook, a API recebe o link que identifica a grade atual do usuário e é responsável por abrir uma janela onde o usuário pode autorizar o compartilhamento do link em seu perfil do Facebook.

A inclusão da API é feita por um único script em JavaScript que executa assincronamente e realiza as ações necessárias sobre elementos HTML com classes específicas para configurar corretamente os botões anteriormente descritos.

4.3.2 Google Analytics

O Analytics é uma ferramenta da Google que, através de código JavaScript inserido no HTML das páginas web, monitora-as e gera diversas estatísticas como total de visualizações da página, países e cidades onde os acessos foram realizados, sistema operacional e navegador utilizados para acessar a página, entre outras. Essa coleta de dados foi mantida na nova versão do MatrUSP e os dados da página antiga foram usados como base para tomadas de decisão durante o desenvolvimento.

Como exemplo, optamos por desenvolver a nova página com recursos que sejam totalmente compatíveis com os navegadores Chrome, Firefox, Safari e Internet Explorer (IE) + Edge que representam respectivamente 77,63%, 10,32%, 8,35% e 1,93% de todos os acessos ao MatrUSP. Ainda, optamos por manter compatibilidade total somente com as versões 9, 10 e 11 do Internet Explorer, uma vez que acessos através dessas versões totalizam 82.89% dos acessos pelo IE e a compatibilidade de funções JavaScript e CSS com as versões anteriores a essas é limitada ou inexistente.

4.4 Pesquisa de boas práticas de desenvovimento

4.4.1 JavaScript

Ainda antes de começarmos a desenvolver, todavia, utilizamos um tempo em busca de como executar esta tarefa da melhor forma possível e procuramos em diversos lugares[7], boas práticas e técnicas para o desenvolvimento em Javascript, a linguagem predominante no nosso código. Também encontramos formas de estruturar o CSS[8] para facilitar sua compreensão para nós mesmos e para aqueles que vierem a ter contato com ele.

Algum exemplos de boas práticas dentro destas duas linguagens, quando consideramos o SASS, são utilizar nomes de variáveis e métodos que sejam expressivas: queremos que elas expliquem o máximo possível o motivo de sua existência. Também buscamos manter o código consistente em relação a pequenos detalhes como o espaçamento entre parêntesis e onde abrir e fechar chaves.

Pela nossa pesquisa, apesar de parecerem coisas pequenas, quando todas elas são levadas em consideração, o resultado final é um código muito mais agradável de se ler e mais fácil de compreender.

4.4.2 Git

Assim como no JavaScript, decidimos que seria bom para o projeto pesquisar por referências de como utilizar o git da melhor forma possível, preparando-o para futuras contribuições, assim como facilitando o nosso próprio entendimento. O resultado desta busca foi relevante, principalmente, na forma como escrevemos nossos commits.

4.5 Documentação

4.5.1 JsDoc

Optamos por criar uma documentação que abrangesse toda a base de código e, para isso, optamos por utilizar a ferramenta JsDoc[9]. Procuramos diversas opções de como documentar o código e, após nos depararmos com uma análise comparativa entre as principais alternativas[10], optamos por utilizar o JsDoc por parecer o mais robusto e completo. Para gerar a documentação, fez-se necessário comentar com um

[7] http://jstherightway.org/

[8] https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Writing_efficient_CSS

[9] http://usejsdoc.org/

[10] http://www.fusioncharts.com/blog/2013/12/jsdoc-vs-yuidoc-vs-doxx-vs-docco-choosing-a-javascript-documentation-generator/

cabeçalho cada uma das funções do código, colocando ali as respectivas descrições. Após isto, processamos a base de código com a ferramenta e, finalmente, temos um website com a documentação pronta[11].

4.5.2 GitHub Wiki

Além da documentação do código em si, mostrou-se interessante criar um ambiente onde pudesse ser exposto um pouco da ideologia por trás do método de desenvolvimento escolhido e, também, os porquês sobre as decisões de design, tanto de software quanto gráfico.

Para este fim, foi utilizada a Wiki[12] do repositório no GitHub[13]. Nela, foi possível estruturar as opiniões sobre as diferentes áreas e perspectivas do projeto, além de se revelar um espaço propício para colocarmos indicações sobre como se envolver no desenvolvimento do MatrUSP, uma vez que estamos tratando de um código aberto.

4.6 Objetos e Classes

Escolhemos utilizar Programação Orientada a Objetos (POO) para facilitar nossa visão do sistema e abstração do problema em questão. Optamos por separar e, em grande medida, encapsular conceitos e funcionalidades, o que gerou, além de outros benefícios, um ganho na compreensão do código como um todo.

O núcleo do sistema MatrUSP, desconsiderando funcionalidades periféricas, consiste em agrupar disciplinas e mostrar graficamente ao usuário os horários das aulas de uma determinada combinação de turmas. Para realizar esta tarefa, criamos classes de objetos que representam disciplinas, turmas, aulas e uma outra representando uma combinação. As combinações representam todas as possíveis permutações dos horários de todas as disciplinas incluídas em um plano.

No nosso sistema é possível a construção de até três planos, ou seja, é possível incluir disciplinas em um plano sem afetar os demais. No nosso código, introduzimos uma classe para os planos permitindo tratá-los de forma similar. Há ainda uma classe para representar o estado, que será descrito na próxima seção e outra para a interface gráfica. Esta última é responsável por encapsular a maior parte da manipulação dos objetos gráficos.

4.7 O Estado

O código original, de certa forma já baseava-se no conceito de estado global para representar as disciplinas e turmas selecionadas em um determinado instante de tempo. Além de nos espelharmos nessa abordagem, nos inspiramos em um framework javascript chamado Redux[14], que tem como ideia fundamental a existência de um estado que centraliza a dinâmica da aplicação: qualquer mudança que aconteça atualiza o estado, que por sua vez computa as alterações necessárias como consequência e propaga-as. Fora isso, a implementação destas ideias se mostrou bastante fácil, uma vez que já tínhamos um alto nível de encapsulamento.

O estado, então, é um objeto que engloba todos os elementos conceituais citados na Subseção 4.6, além de manter referências aos objetos HTML sendo apresentados graficamente aos usuários, em vários casos através de objetos das classes previamente apresentadas. Ampliamos o conceito de estado em relação à base de código antiga para abranger completamente o status do sistema em um determinado instante de tempo, isto é, armazenar todas as informações relevantes para reproduzi-lo.

Desta forma, conseguimos manter todo conteúdo relevante centralizado, tornando muito mais fácil a modificação do sistema, sua manutenção, e sua flexibilidade, por exemplo, quando desejamos acessar informações para diferentes funcionalidades. Além disso, o acesso a diferentes informações torna-se mais metódico e consistente independente do tipo de dado de interesse, o que por outro lado ajuda na compreensão e desenvolvimento do código.

[11] http://bcc.ime.usp.br/matrusp/js/docs/
[12] https://github.com/matrusp/matrusp/wiki
[13] https://github.com/matrusp/matrusp
[14] http://redux.js.org/

5 Código

Como consequência e, em alguns casos, em paralelo a toda a pesquisa e às decisões descritas acima, passamos a desenvolver o novo código do MatrUSP. Como é possível perceber pelo conteúdo apresentado até agora, o código desenvolvido se apoia bastante no conceito de estado e, por conta disso, boa parte do tempo utilizado na programação esteve ligado à implementação dele.

5.1 Árvore de estado

Construímos uma classe chamada "State" a partir da qual criamos um único objeto que representa o estado do sistema, seguindo o design pattern chamado singleton[15]. Parte do objeto é inicializado dentro do construtor e o resto é delegado a um método da classe chamado "load", que apenas inicializa os demais campos se receber um objeto JSON que represente um estado salvo. Ainda no construtor, são criadas referências a objetos HTML que já existem no arquivo estático, os quais fazem parte do conjunto de interfaces que permitem uma mudança de estado, como os botões de upload e download de um JSON e os botões de mudança de combinação.

O estado possui também um vetor com três planos. Estes, são definidos pela classe "Plan", que é a mais complexa da base de código, principalmente por ser ela a responsável pelo cálculo das possíveis combinações.

Um plano, assim como o estado, possui referências a elementos HTML que existem no arquivo estático, mas neste caso cada um refere-se a uma das três abas, onde é possível selecioná-lo ou executar uma de suas opções: limpá-lo, excluindo todas as suas disciplinas, ou copiá-lo para um outro plano. Além disso, ele pode ser inicializado com um objeto JSON de um plano salvo, que pode ser transmitido pela classe State e, nesse caso, é ajustado de acordo para clonar esse plano antigo. Existem ainda atributos que fazem referência à combinação selecionada em um instante de tempo, isto é, à combinação que está sendo apresentada graficamente ao usuário.

Toda vez que ocorre um dos seguintes casos as combinações são recalculadas: inclusão ou exclusão de uma disciplina, mudança de prioridade entre disciplinas, seleção ou omissão de uma turma dentro de uma disciplina, seleção ou omissão de uma disciplina em si, e o carregamento de um plano salvo. O algoritmo que encontra as possíveis combinações leva em consideração todas as permutações de turmas por disciplinas e, para cada permutação, testa se ela é uma combinação viável, isto é, se não há conflito de horários. Este algoritmo será descrito em maior profundidade na Subseção 6.1.

Dentro do objeto da classe "Plan" existe um vetor de combinações. Cada combinação, criada a partir da classe "Combination", representa a escolha de uma turma para cada disciplina daquele plano e, para expor tais escolhas, têm um vetor com referências para cada uma dessas turmas. Estas referências apontam para representações de turmas que serão descritas mais abaixo.

Um plano possui também um vetor de disciplinas, que são instâncias da classe "Lecture". Pensamos em utilizar o nome "Class", mas pela possível confusão com classes de POO ou CSS optamos pelo outro. Diferente dos anteriores, os objetos dessa classe possuem vínculo com elementos HTML criados dinamicamente e, portanto, invocam a criação destes elementos dentro de seus construtores e guardam referências para eles. Lectures, como se pode esperar, também armazenam o nome da disciplina, seu código e um atributo indicativo que informa se a disciplina está selecionada ou não.

Uma disciplina, por sua vez, guarda um vetor de turmas, que são objetos da classe "Classroom". Cada turma está ligada a seus objetos HTML específicos, o que é feito de forma semelhante às disciplinas. Dentro de uma turma, temos basicamente as informações sobre o nome dos professores e o código da turma.

Por fim, cada turma possui um vetor de horários de aula, objetos da classe "Schedule", que guardam apenas as informações sobre o dia da semana e horários de início e fim.

Com exceção do estado, que é um singleton, todos os objetos dessas classes possuem referência aos seus "pais", por exemplo, um objeto de "Classroom" sabe a qual objeto de "Lecture" ele pertence, assim como este sabe qual instância de "Plan" o gerou, que é a qual ele pertence.

Ademais, citamos onde e quais referências a objetos HTML são criadas, mas ainda existe a interação entre eles e as instâncias das classes aqui descritas. Para permitir a interação do usuário e responder às mudanças de estado que elas implicam, estas instâncias criam ouvintes para certos tipos de interação com

[15] https://pt.wikipedia.org/wiki/Singleton

elementos HTML e vinculam a elas métodos de callback. Quando um evento acontece e sinaliza um ou mais destes ouvintes, os métodos vinculados são executados e, então, o estado pode reagir adequadamente.

5.2 Exemplo

Um estado pode, e em geral possui diversos objetos em sua árvore. Para exemplificar de forma mais clara e simples, apresentamos, aqui, um diagrama de classes:

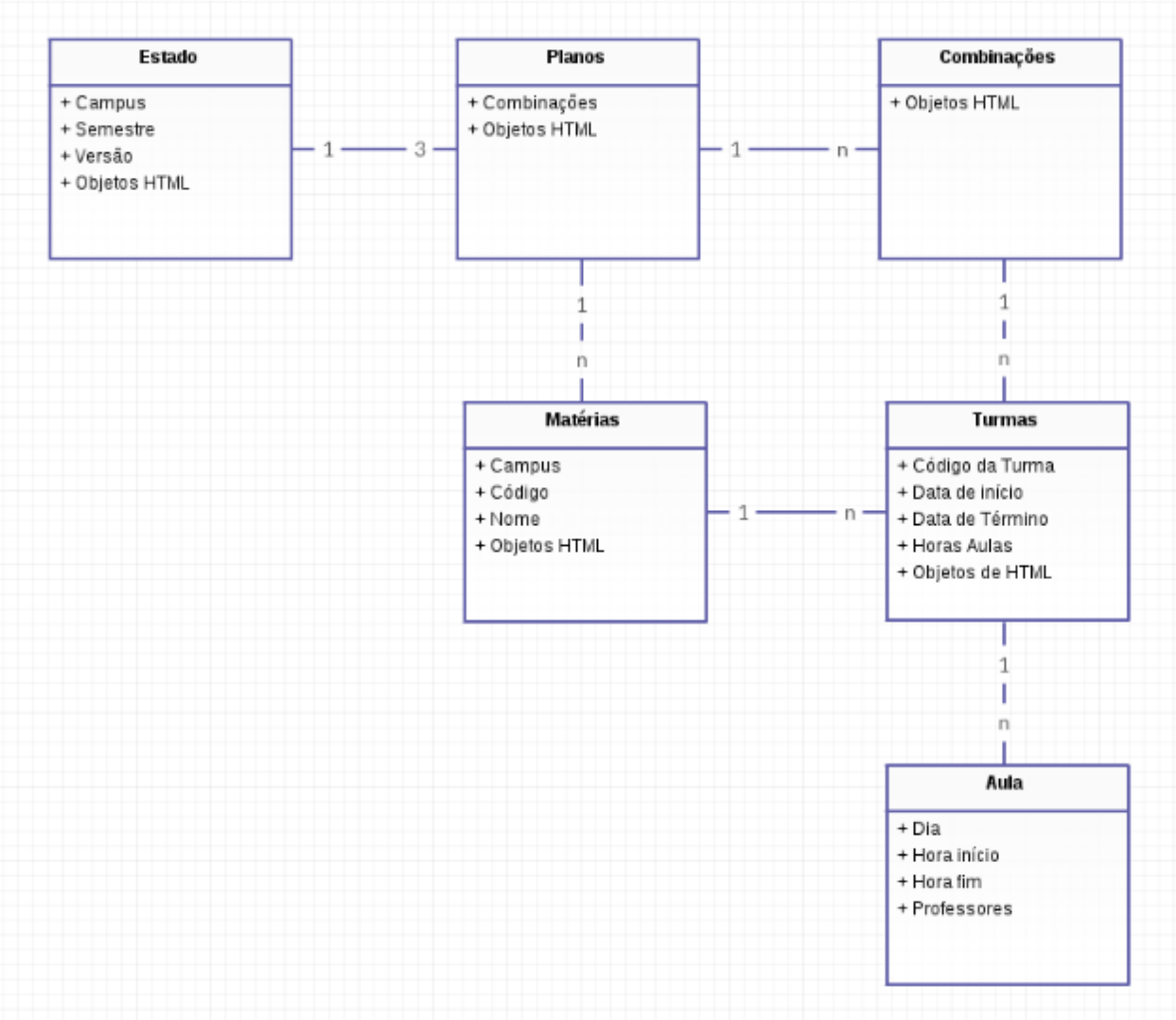

Figura 3: Diagrama de Classe

5.3 Bibliotecas

Além das classes que representam o estado, as quais definem o núcleo do aplicativo, criamos uma outra classe, seguindo o paradigma de singleton, com a intenção de encapsular toda a lógica que envolve elementos da parte de interface gráfica exclusivamente. Chamamos esta classe de "UI" e é a partir dela que geramos, por exemplo, elementos HTML que representam os horários de aula, ou ainda os elementos que contêm os nomes dos professores para uma determinada turma.

Vimos também a necessidade de utilizar certos trechos de código em diversas partes da base de código e, por isso, optamos por construir uma pequena biblioteca personalizada com os métodos que apareciam mais de uma vez ou que se apresentassem genéricos o suficiente e com potencial de serem reutilizados.

6 Funcionalidades Desenvolvidas

6.1 Montar grade

A principal funcionalidade do MatrUSP é mostrar graficamente ao usuário como os horários de aulas de diferentes disciplinas se encaixam ao longo de uma semana. Para realizar esta tarefa, construímos dois elementos básicos: um painel para representar todos os horários de segunda-feira a sábado em formato de grade, e um objeto representando, visualmente, o período de uma aula: dentro do painel, um retângulo posicionado na coluna do dia da semana daquela aula, abrangendo desde a linha do horário de início até a linha do horário de fim da aula. O primeiro deles, o painel, faz parte da estrutura HTML estática, enquanto que as aulas são elementos dinâmicos associados a objetos da classe "Schedule".

Além disso, ao lado deste painel, há uma lista para mostrarmos ao usuário sua atual seleção de disciplinas. Utilizamos duas formas para indicarmos a qual disciplina uma certa aula pertence. Primeiro, toda aula possui no centro de seu retângulo o código da disciplina. Ademais, criamos um esquema de cores no qual toda aula possui a mesma cor de fundo da disciplina à qual ela pertence.

Com as disciplinas já selecionadas, o MatrUSP calcula e apresenta as diferentes combinações possíveis. Para tanto, foi necessário criar um algoritmo que encontrasse todas as possibilidades, onde um conjunto de turmas é uma combinação viável se, para todo par de aulas, não há sobreposição de horários entre elas. Então, criamos métodos de teste de inviabilidade e, para toda potencial combinação, testamos se os pares de aulas são viáveis ou não. Quando todos os pares são viáveis, adicionamos tal combinação a um vetor de combinações, como descrito anteriormente dentro da árvore de estado.

Esta solução para o problema é de ordem exponencial, ou seja, se aumentarmos linearmente o número de disciplinas (considerando disciplinas com mais de uma turma), aumenta-se exponencialmente o tempo que leva para testar todas as possibilidades. Entretanto, na prática, esta característica não afeta de forma significativa o desempenho do sistema, uma vez que a quantidade de disciplinas adicionadas é baixa comparada ao nível de processamento dos navegadores modernos.

Para facilitar a navegação do usuário pelas combinações, criamos uma área na página com miniaturas de painéis de grade horária, cada um populado com miniaturas das aulas. Assim, procurar uma combinação que melhor se encaixe nos interesses de quem está utilizando o sistema fica mais fácil. Por exemplo, é muito fácil notar quando todas as aulas são à noite ou, em outro caso, são todas pela manhã.

Em alguns casos, uma disciplina pode conter apenas turmas que entrem em conflito com as outras disciplinas em todas as possíveis combinações. Nesse caso, impossibilitamos a opção de seleção da disciplina enquanto não houver uma combinação viável que a inclua. Isto pode acontecer de três formas: removendo alguma outra disciplina, desselecionando um disciplina com maior prioridade, ou ainda mudando a prioridade desta disciplina.

Nos dois últimos casos, a dinâmica de busca por combinações viáveis pauta-se no conceito de prioridade de disciplinas. Este conceito é aplicado de forma natural de acordo com a ordem em que as disciplinas foram incluídas pelo usuário: a última a ser incluida tem a menor prioridade. Para permitir a mudança deste cenário, construímos botões que fazem uma disciplina pular um degrau para baixo ou para cima na lista e, consequentemente, mudar sua prioridade no cálculo de combinações.

Finalmente, o último item incluso nesta funcionalidade é a opção de selecionar e desselecionar uma disciplina e cada uma de suas turmas independentemente. Isto permite que o usuário exclua do cômputo de combinações uma ou mais turmas específicas, por exemplo, por não querer estudar à tarde, ou ainda, permite que ele exclua uma disciplina inteira, por exemplo, para testar as opções de grade horária se ele desistir de cursar aquela disciplina.

6.2 Busca por Disciplinas

A seleção de disciplinas é uma parte importante do sistema, é nela que ocorre o intercâmbio de informações entre a nossa base de dados e a interface gráfica, sendo possível localizar as matérias existentes num semestre atual.

Todos os caracteres inseridos são transformados em caracteres simples, sem acentuação e minúsculos, assim, independente do modo como são inseridas essas informações, é possível realizar a busca.

Para cada letra inserida no campo de busca, uma nova expressão regular é criada e consequentemente inserida em um vetor, ela é formada por todos os caracteres inseridos até o momento acrescido do novo caractere. Após a sua composição, esse vetor é utilizado para procurar possíveis candidatos na base de dados, dando uma nota para os que mais se aproximam dos caracteres inseridos e ordenando-os.

Essa nota é gerada pela soma de dois critérios, o primeiro procura por strings que coincidem inteiramente com a expressão regular procurada, esse é o critério que dá a maior nota, o segundo procura por subsequências das strings que condizem com a regex procurada. Quando os caracteres digitados pelo usuário corresponde, exatamente, com o nome ou o código de uma disciplina, ou seja a expressão regular é igual ao nome da matéria ou ao seu código, o único item retornado pela lista é essa disciplina.

6.3 Identificador no servidor

Criamos a possibilidade de um usuário salvar uma grade gerada por ele no servidor, para isso criamos o identificador. Para um usuário salvar a sua grade, ele deve escolher um nome que será atribuído a esse identificador, ou seja, o nome escolhido pelo usuário será o nome do arquivo salvo no servidor. Esse arquivo contém o estado transformado em um JSON.

Em JavaScript não existe um método simples para copiar um objeto, uma vez que uma atribuição é a criação de uma referência direta para o objeto. Desse modo, foi necessário a criação de funções que percorrem o estado copiando o que é relevante e removendo eventuais atributos que não seriam necessários para o arquivo JSON.

6.4 Compartilhar

O mecanismo responsável pelo compartilhamento tanto pelo Facebook quanto pelo e-mail é o identificador, quando um usuário decide compartilhar a sua grade, o nosso programa cria um identificador que é embutido na URL e pode ser enviado para outras pessoas. Se o usuário escolher "Enviar por E-mail", o link preparado com o identificador é exibido para o usuário copiar e colar onde desejar, se a escolha for "Compartilhar pelo Facebook" o link é direcionado à API do Facebook, que realiza os procedimentos necessários para abrir uma janela onde o usuário poderá compartilhar o link do MatrUSP com alguma mensagem personalizada, ou vazia, diretamente em seu perfil na rede social.

6.5 LocaStorage

É muito confortável entrar num site e não ter a necessidade de preencher todas as informações novamente. Essa ferramenta permite que toda a vez que um usuário acesse o site pelo mesmo navegador o último estado seja carregado, sem a necessidade de colocar novamente as informações.

Assim como o Identificador, ela utiliza as informações salvas no estado para gerar uma string que será carregada no próximo acesso.

6.6 Download/Upload de .json

A possibilidade de baixar a grade horária construída através do MatrUSP em um arquivo .json foi mantida, porém o formato do conteúdo escrito em json foi levemente modificado entre as versões do sistema. Todas as informações armazenadas anteriormente foram mantidas, porém agora o arquivo armazena uma cópia exata da árvore de estados do sistema num dado momento do download.

Assim, a possibilidade de carregar arquivos .json no sistema para exibir grades anteriormente geradas pelo mesmo também foi mantida, e uma nova árvore de estados é atribuída a partir da análise da árvore contida no arquivo. Deste modo, não há compatibilidade para carregar, ler e exibir grades armazenadas em arquivos gerados pelo antigo MatrUSP na nova versão do sistema.

6.7 Download do PDF

Uma preocupação que tivemos foi providenciar um modo de consulta off-line à grade organizada pelo usuário, ou seja, uma forma que não requeresse o uso de conexão com a internet para consulta-lá. A melhor

solução nos pareceu ser exportar o quadro exibindo os dias da semana e as matérias junto com uma legenda com código, turma, professor e nome das matérias, para que o usuário possa consultar seus horários com agilidade. Com isso, surge um modelo de visualização que, além de existir digitalmente como PDF, pode ser impresso sem riscos de modificar sua formatação, algo que o padrão PDF garante.

Para desenvolver essa funcionalidade incluímos três bibliotecas ao nosso projeto:

- HTML2Canvas: Utilizada para gerar um "screenshot" de elementos DOM
- JSPDF: Utilizada para gerar um arquivo .pdf
- AutoTable: Utilizada para gerar uma legenda no arquivo .pdf

Para gerar o arquivo .pdf, as funções contidas no arquivo "pdf.js" atuam da seguinte maneira:

- Uma imagem da grade horária é produzida a cargo da biblioteca HTML2Canvas
- A imagem é dimensionada e colocada no topo de um documento pdf, criado pela biblioteca JSPDF
- A variavél que guarda o estado do sistema é percorrida e as informações dos horários em exibição são extraídos para a biblioteca AutoTable gerar a legenda e posicioná-la logo abaixo a imagem
- Após esse processamento ser finalizado o usuário é requisitado para fazer o download do arquivo

6.8 Exportar grade horária

6.8.1 iCalendar

Atualmente existem diversas aplicações para ajudar os usuários a gerenciar seu tempo como Google Calendar, Yahoo Calendar e o Outlook Calendar, este último desenvolvido pela Microsoft. Esses aplicativos costumam exibir calendários onde os usuários configuram eventos, com data de início e fim, horário de início e fim, recorrência, título, sumário, entre outros componentes, e podem visualizar essa organização graficamente. Muitos desses programas utilizam arquivos no formato iCalendar, com a extensão '.ics', para definir tais eventos em ações de exportação e importação. Uma das funções do novo MatrUSP é exportar as tarefas incluídas pelo usuário no sistema para qualquer uma dessas aplicações gerenciadoras de tempo através de arquivos no formato iCalendar.

O formato iCalendar for definido pela Internet Engineering Task Force (IETF), uma organização que desenvolve e promove voluntariamente normas aplicáveis à internet e publica "Requests for Comments" (RFCs). RFCs, por sua vez, são documentos que contém notas técnicas e organizacionais sobre a internet e são publicados por memorandos. O RFC que define as normas atuais do formato iCalendar é o RFC5545.

O RFC5545, no qual nos basamos durante a elaboração desse projeto, foi publicado em Setembro de 2009 e substituiu as definições explicitadas no RFC2445, de Novembro de 1998, defininido a seguinte estrutura básica para a definição de um arquivo iCalendar:

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//hacksw/handcal//NONSGML v1.0//EN
/* corpo do arquivo */
END:VCALENDAR
```

Cada linha nesse tipo de arquivo descreve uma propriedade, atributos que aplicam-se ao objeto de calendário como um todo. No corpo do arquivo são definidas componentes, grupos de propriedades que expressam uma semântica particular de calendário, e podem especificar eventos, "to-dos", entradas de diário, informações sobre fusos-horários, informações de tempo "livre/ocupado" (tradução livre de free/busy) ou um alarme. Optamos por definir cada horário exibido ao usuário no MatrUSP como uma componente "VEVENT", no arquivo iCalendar, que especifíca uma componente de evento.

```
BEGIN:VEVENT
DTSTART;TZID=America/Sao_Paulo:20130226T080000
DTEND;TZID=America/Sao_Paulo:20130226T094000
RRULE:FREQ=WEEKLY;UNTIL=20130629T235959;BYDAY=TU
DTSTAMP:20161102T181633743Z
UID:20161102T181633743Z-MAC0110T41DTU@localhost
SEQUENCE:0
STATUS:CONFIRMED
SUMMARY: Aula de MAC0110 - Introducao a Computacao
TRANSP:OPAQUE
END:VEVENT
```

A definição de eventos deste modo é muito conveniente uma vez que a sintaxe iCalendar nos permite definir para cada aula diversas propriedades úteis como data e horário de início da primeira ocorrência da aula (DTSTART), data e horário de fim da primeira ocorrência da aula (DTEND), regra de recorrência da aula (RRULE, no exemplo: toda terça até o fim do semestre) e tema do evento (SUMMARY).

No MatrUSP, a montagem de um arquivo desse tipo fica a cargo das funções descritas no arquivo "icalendar.js". Quando o usuário clica no botão "exportar para arquivo .ics", acessível através de um clique no ícone de calendário no canto superior esquerdo da página o arquivo é preparado da seguinte maneira:

1. Os horários das turmas escolhidas pelo usuário são percorridos um a um;
2. Para cada horário distinto é criado um componente de evento;
3. Para criar a componente de evento funções auxiliares recuperam e formatam as seguintes informações adequadamente:
   - Data de início das aulas da classe: para as propriedades DTSTART e DTEND
   - Data de fim das aulas da classe: para a propriedade RRULE
   - Horário de início da aula: para a propriedade DTSTART
   - Horário de fim da aula: para a propriedade DTEND
   - Dia da semana da aula: para a propriedade RRULE
   - Dia e horário atual: para a propriedade DTSTAMP
   - Identificador do evento: para a propriedade UID
   - Código e nome da matéria: para a propriedade SUMMARY
4. As seguintes propriedades são estáticas para todas as componentes de evento construídas e completam a definição das componentes:
   - SEQUENCE:0
   - STATUS:CONFIRMED
   - TRANSP:OPAQUE
5. Todas essas propriedades são montadas em linhas distintas de uma string que juntamente com as propriedades obrigatórias 'BEGIN:VCALENDAR', 'VERSION', 'PRODID' e 'END:VCALENDAR', devidamente ordenadas e descritas, são codificadas pela função "encodeURIComponent()", nativa do JavaScript
6. A string codificada é imcumbida ao atributo 'href' de um elemento HTML 'a' e o download do arquivo é forçado através de um evento 'click' disparado assim que todo esse processamento é concluído

6.8.2 API do Google

O Google Calendar é uma aplicação web de gerenciamento de tempo criada em 2006 pela Google. Para adicionar eventos nesse serviço da Google o usuário pode optar por utilizar a interface gráfica do aplicativo, selecionar um arquivo .ics com a descrição de um ou mais componentes ou incluir os componentes usando JavaScript diretamente no calendário padrão do usuário dada a autenticação da conta Google e a autorização do usuário para o MatrUSP. Optamos por desenvolver essa funcionalidade que facilita a importação exclusivamente para a plataforma Google Calendar pois a sua API é a melhor documentada entre as de serviços semelhantes.

Para configurar a API foi necessário criar um projeto referente ao MatrUSP no console de desenvolvedor Google e gerar uma credencial, com a credencial configuramos ainda os domínios que podem fazer requisições. Com a credencial devidamente configurada para verificar se o usuário concedeu ou deseja conceder à credencial do MatrUSP o direito de modificar os calendários do usuário são chamadas as versões modificadas das funções descritas no guia de referência rápido[16] à API em JavaScript. Para inserir os eventos no calendário a variável de estado é percorrida e para cada horário em exibição são extraídas as seguintes informações: data de início da primeira ocorrência da aula, data de fim da primeira ocorrência da aula, horário de início, horário de fim e recorrência da aula, essas informações são inseridas em um objeto de evento (esse formato é definido pela API do Google Calendar). Para cada evento é feita uma requisição através da API que envia os dados para o calendário padrão da conta Google do usuário.

[16] https://developers.google.com/google-apps/calendar/quickstart/js

7 Design

Se compararmos o design do MatrUSP antigo com qualquer página web com um design moderno, é notável a discrepância entre os seus elementos: cores, bordas, alinhamento, como os objetos estão dispostos e como eles interagem uns com os outros. Nesse sentido, tentamos repensar o desenho da aplicação com o intuito de deixá-lo mais bonito, atraente, mais claro e fácil de utilizar.

7.1 Inspiração

Sabemos que a Apple Inc. é uma empresa com excelente reputação de design e, como nosso sistema se baseia em grande parte na apresentação de eventos (aulas) em uma grade horária semanal, achamos uma boa escolha nos basearmos no programa de gerenciamento de calendário da Apple.

Acreditamos que conseguimos trazer alguns elementos que ajudaram o MatrUSP a ficar mais intuitivo e prazeroso de utilizar, entre eles, a paleta de cores. Não extraímos diretamente as cores utilizadas no calendário da Apple, mas sim o tom delas, o que gerou uma aparência bem mais suave à aplicação. Além disso, no todo, o branco e tons claros de cinza foram bastante empregados ajudando nesta suavização.

Também chamou nossa atenção a apresentação minimalista no design da Apple, o que serviu de grande inspiração nas decisões sobre quais componentes incluir no nosso programa. Como resultado, por exemplo, excluímos boa parte das bordas que existiam no design antigo, o que, ao nosso ver, deixou a nova página muito mais compreensível.

7.2 Princípios de IHC

As decisões de design não foram arbitrárias, pelo contrário, elas se basearam em conceitos de Interação Humano-Computador (IHC) como o CRAP: Contraste, Repetição, Alinhamento e Proximidade. Em mudança à versão original, optamos por agrupar os elementos em 4 partes.

A grade horária, que é extremamente importante, ganhou maior destaque ficando centralizada e ocupando a maior área do plano de visão do usuário. Já as informações sobre as disciplinas ficaram em ao lado dela, ganhando bastante relevância, mas agora, reunindo todas as informações sobre as disciplinas no mesmo lugar. Na versão original elas estavam divididas, com as informações sobre as turmas aparecendo em um lugar separado do nome e código da disciplina.

Em ordem decrescente de relevância, ainda separamos a área de funcionalidades externas, agrupadas ícones e, finalmente, uma seção de rodapé com as opções de nos contactar e envolver-se no projeto.

Por toda a interface gráfica, tentamos garantir que não houvessem surpresas (ruins) da perspectiva do usuário, isto é, se há algum elemento parecido com outro na página, ambos reagem da mesma forma. Também, se algum elemento incita alguma tipo de ação, então ele responderá àquela ação. Um exemplo disso é um item mudar de cor e/ou forma quando o cursor está em cima dele e, portanto, ele executar alguma coisa quando for clicado.

7.2.1 Comparação entre o antigo e o novo

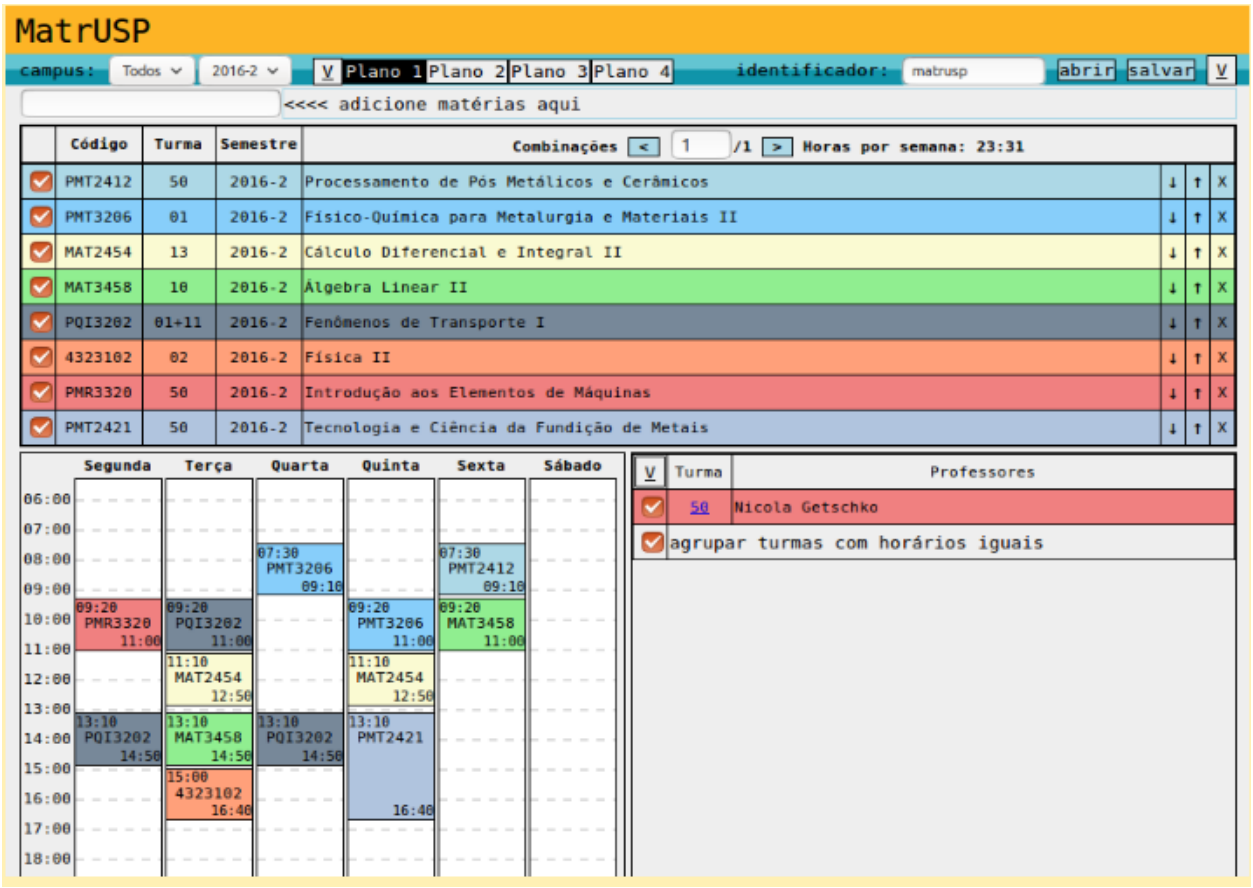

Figura 4: MatrUSP antigo

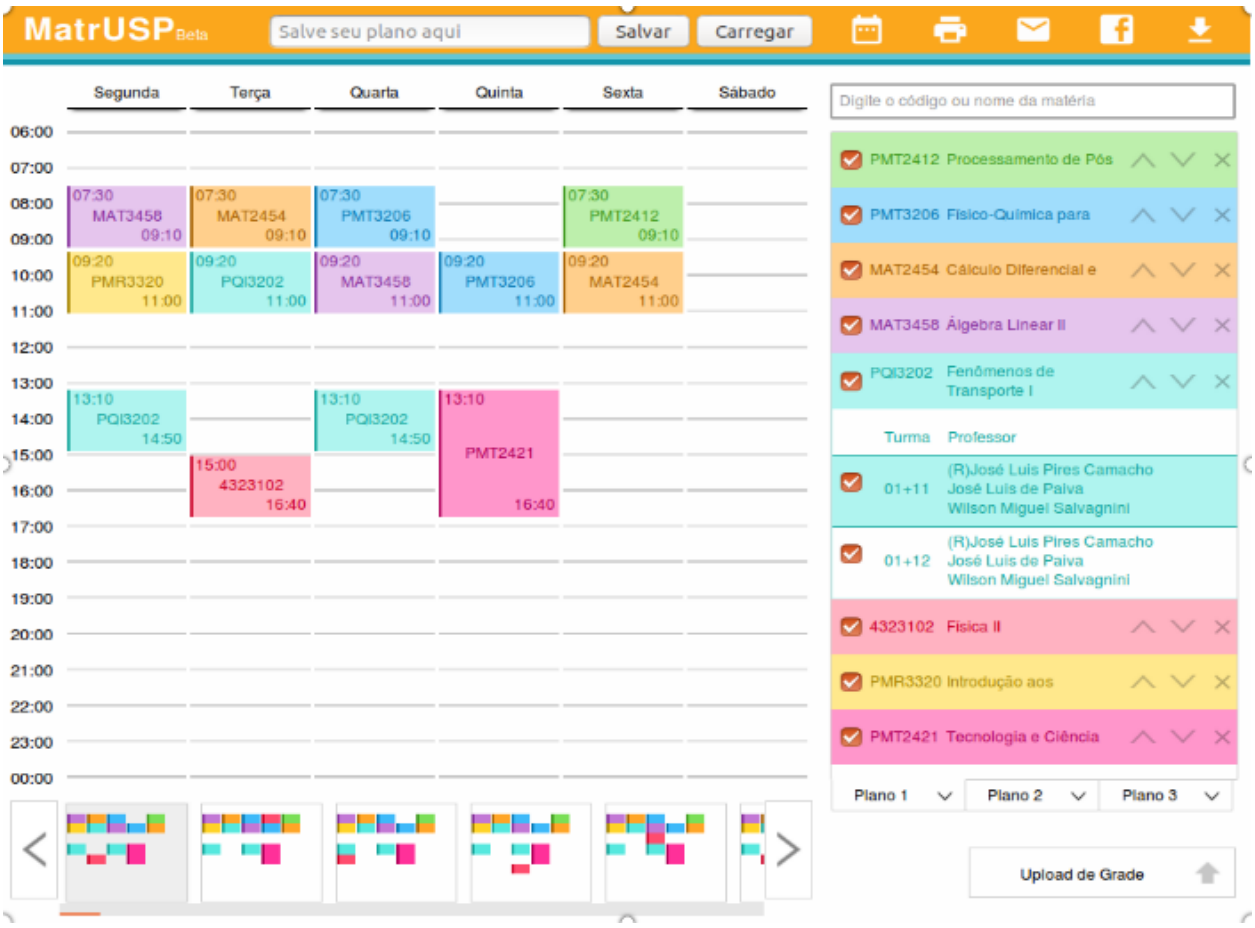

Figura 5: MatUSP novo

7.2.2 Não tão nova funcionalidade

Durante o desenvolvimento, percebemos que nós mesmos, que já possuíamos alguma intimidade com o sistema, desconhecíamos a funcionalidade de computar diferentes combinações para um dado conjunto de disciplinas. Para remediar esta situação, reformulamos a forma com a qual é possível explorar tais combinações.

Antigamente, havia apenas um campo, no topo da página, onde era possível ir de uma combinação para outra. Desenhamos e construímos então uma seção onde é possível olhar miniaturas das combinações existentes e navegar por elas com maior facilidade e um feedback visual, definitivamente, muito maior e melhor.

8 Web Crawler

Web crawler é o nome que se dá a um mecanismo criado para procurar e extrair informações de páginas ou indexar páginas de um website, é um programa de computador que navega pela World Wide Web de maneira automatizada e metódica, sendo utilizado pela maioria dos mecanismos de buscas. Hoje em dia, devido à grande quantidade de informação contida na internet, é uma ferramenta muito comum, entretanto, segundo nossas pesquisas, não existe nenhum web crawler genérico de boa qualidade que pudesse atender às nossas demandas: extrair informações sobre as disciplinas no JúpiterWeb.

Encontramos algumas opções, mas mesmo a melhor delas[17] se mostrou pouco flexível quanto a uma eventual mudança do layout da página alvo. Sendo assim, optamos por manter a parte do código já responsável por esta extração, existente desde a versão anterior do MatrUSP.

Como ainda não existe uma forma prática de extrair estas informações (como, por exemplo, uma API disponibilizada pela USP) essa ferramenta torna-se imprescindível para nós. Todas as informações que disponibilizamos em relação às matéria são adquiridas através deste crawler. Ele navega por algumas páginas do JupiterWeb coletando todas as informações relacionadas às matérias, como o seu código, nome, professores que irão ministrá-la, entre outros.

Contudo, isso apresenta um problema, pois o modo que os crawlers capturam as informações contidas em uma URL é navegando pelas tags HTML dela, sendo assim, se ocorrer uma mudança na sua estrutura, o programa não irá mais funcionar e consequentemente o MatrUSP não terá mais acesso a essas informações até que o código seja adequado ao novo formato do HTML.

[17] https://www.uipath.com/community

9 Conclusão

9.1 Onde Chegamos

Acreditamos que conseguimos fazer grandes melhorias no MatrUSP.Assim como antes, é possível montar uma grade horária com feedback visual dinâmico, sendo muito simples perceber conflitos entre matérias, e podendo adequá-las da melhor forma possível. Além disso, com o novo design é possível, de uma forma muito mais simples, navegar pelas diferentes combinações das turmas das disciplinas selecionadas, o que torna muito simples a escolha da grade horária que melhor se adeque ao próximo período do usuário.

Isso adicionado a novas funcionalidades como a integração com o calendário do google, o download do arquivo .ics (que torna possível a sincronização com outros calendários), a possibilidade de compartilhar a grade tanto por e-mail quanto pelo Facebook, trouxeram uma melhora notável para o sistema. Trazem facilidades para os usuários que antes não existiam.

Também nos preocupamos em realizar melhorias pensando na continuidade do sistema. A decisão de reescrever o código tornou a adesão de novos desenvolvedores muito mais tangível, uma vez que tomamos uma série de decisões para o código ser o mais claro possível, como a escolha de utilizar POO para estruturar o sistema, escolher nome de variáveis e funções que por si só descrevem o seu propósito, buscar a menor repetição possível de código, a criação de uma documentação e uma wiki, entre outras.

9.2 Reflexão sobre as metas

Optar por reescrever o código todo se mostrou algo muito bom, uma vez que tivemos a experiência de construir um produto por inteiro, passando por todos os estágios da criação de um software: desde busca, identificação e desenho dos requisitos e funcionalidades do sistema, até a programação de cada componente e como tornar o código flexível para ser executado em diversos navegadores disponíveis no mercado. Também com essa decisão vieram os problemas em cada uma das etapas, que nos fizeram refletir bastante e, em certos casos, apreciar ainda mais o conhecimento adquirido ao longo da graduação.

Em relação às funcionalidades do sistema antigo, com exceção da possibilidade de adicionar atividades extracurriculares personalizadas, conseguimos implementar todas na versão atual. Entre as novas, que foram sugeridas no começo do trabalho, sabíamos que não teríamos tempo suficiente para a realização de todas, dessa forma avaliamos quais eram, entre elas, as mais importantes para nós a serem implementadas..

Durante todo o processo nos preocupamos em deixar o código o mais claro possível: com a uma alta taxa de coesão, baixo acoplamento e tentando diminuir ao máximo a quantidade de código repetido. Estes foram alguns dos principais motivos para termos optado por manter uma estrutura de classes e objetos. Além disso, nos preocupamos em criar uma documentação e uma wiki no GitHub.

Todas essas medidas foram tomadas com o intuito de deixar o ambiente o mais preparado possível para os próximos desenvolvedores, pois acreditamos que desenvolver o MatrUSP foi uma grande experiência. Mesmo não tendo realizado todas as funcionalidades que desejávamos (já era sabido que não teríamos tempo para realizar todas), conseguimos estruturar o código de uma maneira clara, sendo fácil a implementação de novas funcionalidades, inclusive por pessoas que não estiveram envolvidas no início do projeto.