# Roteiro de exercícios sobre SaaS, parte de GUI, disciplina Engenharia de Software e Sistemas

## Paulo Borba

## Centro de Informática

## Universidade Federal de Pernambuco

Adicione as respostas imediatamente após as perguntas, usando uma outra cor. Leia com calma as instruções, e reflita com cautela sobre as consequências de cada comando. Leia cada questão até o final, e só depois comece a respondê-la.

## Aula 1

1. Para este roteiro e os próximos, use o Linux/Unix ou um sistema derivado, como o Mac OS. Se você só tem acesso a um computador com Windows, instale o [cygwin](https://www.cygwin.com) para obter comportamento similar ao Linux/Unix. Se você não é familiar com *shells*, variáveis de ambiente, etc. é essencial estudar agora; este [tutorial](https://github.com/chrisparnin/EngineeringBasics/blob/master/Shells.md#shells) traz um bom resumo. Qual o comportamento dos comandos **cat** e **touch**? Qual o comportamento do comando **ls | wc**? Qual o significado da barra entre **ls** e **wc** no exemplo anterior?
2. O que são variáveis de ambiente e para que servem? Qual o propósito da variável de ambiente **PATH**? Nas questões a seguir deste roteiro, registre as pastas nas quais serão instalados os programas solicitados. Em seguida a instalação, se necessário, adicione os caminhos dessas pastas na variável **PATH**. No Linux, **$PATH** mostra o valor atual dessa variável, e **PATH=$PATH:caminho\_adicional** adiciona um novo caminho ao **PATH**. No Windows, **set PATH** mostra o valor atual dessa variável, e **PATH=%PATH%;caminho\_adicional** adiciona um novo caminho ao **PATH**. Cole a seguir o valor atual de **PATH** na sua máquina. Para não ter que inicializar manualmente o **PATH** toda vez que entrar em uma máquina, altere o arquivo **.bashrc** (ou similar) que está no seu diretório raiz (em área que você tenha acesso de escrita) e adicione a declaração da variável **PATH** como ilustrado acima. Após alterar o **.bashrc**, digite **source .bashrc**. O que faz o comando **source**?
3. Digitando **npm -version** no terminal de linha de comandos, verifique se o gerenciador de pacotes do [Node.js](https://nodejs.org/en/) está instalado na sua máquina, e cole a seguir a imagem do seu terminal. Acima, **-version** é considerada uma opção de linha de comandos. Para o npm, você pode listar as opções disponíveis chamando **npm -help**, mas vários comandos adotam a mesma convenção ou similar (**-h**). Descubra o que faz o comando **man** do Unix, chame-o para o **npm** e cole a seguir uma imagem do que foi apresentado no seu terminal.
4. Digitando **node --version** no terminal de linha de comandos, verifique se o [Node.js](https://nodejs.org/en/) está instalado na sua máquina, e cole a seguir a imagem do seu terminal. Se o **node** ou o **npm** não estiverem instalados, crie um diretório (como **saas** ou similar) e siga as instruções de instalação seguindo o *link* anterior (**sudo** ou autorização de acesso pode ser necessária). Se for instalar nas máquinas do CIn ou outros ambientes com restrições de acesso, opte por instalações locais (não globais), e use ferramentas auxiliares (como o [CInInstala](https://gist.github.com/t0rr3sp3dr0/af2e4eafd721d365c7bb6db09919eff9)) que ajudam a instalar software nas máquinas do CIn. Depois de instalar, lembre de atualizar o **PATH**.
5. Digite **ng version** no terminal de linha de comandos e verifique se [Angular](https://angular.io/guide/quickstart) está instalado na sua máquina. Cole a seguir a imagem do seu terminal. Se Angular não estiver instalado, instale digitando **npm install -g @angular/cli**. Se você não tiver permissões adequadas de escrita na sua máquina, você talvez tenha dificuldade de instalar o angular globalmente; se for o caso, não use a opção "-g" (nesse caso de instalação local, a instalação será feita no subdiretório **node\_modules** de onde você chamou o comando de instalação; no Windows, verifique a pasta **.bin** de **node\_modules**, e no Linux verifique o diretório com o mesmo nome do programa instalado). Se for relatada alguma incompatibilidade entre a versão de Node.js e Angular, siga as [recomendações](https://hosting.review/tutorial/how-to-update-node/) para atualizar o Node.js.
6. Se você ainda não criou um *clone* do [repositório](https://github.com/pauloborba/teachingassistant) do projeto do assistente de ensino, crie agora (por exemplo, em um diretório chamado **teachingassistant**). Selecione o primeiro *commit* do projeto e crie um *branch* (por exemplo chamado **SaaS1**) apontando para o mesmo, e solicite a limpeza das áreas de armazenamento local e *stage*, digitando os comandos a seguir:

**git checkout -b SaaS1 e670da4c7dda011225d4a6942149cfdcde6e8667**

**git reset --hard e670da4c7dda011225d4a6942149cfdcde6e8667**

Verifique se o conteúdo do seu diretório local é o mesmo do

[primeiro commit](https://github.com/pauloborba/teachingassistant/commit/e670da4c7dda011225d4a6942149cfdcde6e8667) na história do projeto; basta clicar no terceiro ícone à

direita desse *commit* na página que lista os *commits* do projeto no GitHub

([*teachingassistant*](https://github.com/pauloborba/teachingassistant/commits/master)). Se houver diferenças, apague os novos arquivos e

diretórios, e verifique o status do seu repositório git. Verifique agora como

está a história do seu repositório local e cole a seguir a imagem da mesma

no seu terminal.

1. No diretório onde está o seu *clone*, crie a versão inicial do seu projeto Angular digitando (use **ta-gui**; poderia ser qualquer outro nome mas traria dificuldade no processo de *cherry-pick* que faremos mais adiante)

**ng new ta-gui**

Para a pergunta*Would you like to add Angular routing?* responda*Yes.* Para

a pergunta *Which stylesheet format would you like to use?* responda *CSS.*

Cole abaixo a imagem do seu terminal.

1. Coloque em execução a versão inicial do seu sistema digitando, no diretório **ta-gui**, o seguinte comando

**npm start**

Veja o sistema em execução digitando a URL abaixo no seu browser:

[**http://localhost:4200**](http://localhost:4200)

Cole a seguir a imagem da página do seu *browser* com o sistema em

execução.

1. Se tudo estiver OK, dê logo um *commit* com as mudanças feitas até este momento para criar a versão inicial do seu projeto Angular a ser usado nos roteiros de SaaS. Certifique-se de que arquivos auxiliares e temporários, como os em **node\_modules**, não serão incluídos no *commit*. Dessa forma não poluímos o repositório e simplificamos a tarefa de quem for revisar as nossas contribuições. Dando *commits* para pequenas mudanças e evoluções na configuração do projeto pode ser essencial caso depois algo deixe de funcionar. Será bem mais fácil identificar que mudança causou o problema, e revertê-la, se necessário. Para isso, é também importante escolher para o seu *commit* uma mensagem bem representativa das mudanças gravadas nele. Envie o *commit* criado para um repositório remoto (um *fork* do projeto original, ou um novo repositório criado no GitHub), de forma que você possa continuar seu trabalho em outro computador na próxima aula, ou mesmo se quiser revisar algo em casa. Essas são atividades que devem ser feitas pelo menos ao final de cada aula da disciplina. Cole a seguir a imagem do seu terminal mostrando a nova história do projeto (use **git log --branches --graph**).
2. Opcionalmente, se você quiser trabalhar de casa (**favor não escolher essa opção para as aulas de laboratório**) e não tiver em casa computador com configuração necessária para instalar o Angular e o Node.js, cadastre-se em uma IDE *on-line* para Angular: <https://plnkr.co> ou <https://stackblitz.com>, que tem umas funcionalidades mais interessantes do que a primeira, mas que talvez seja menos estável. Crie um novo projeto (Angular, não AngularJS) nesse ambiente, e em seguida altere esse projeto para que ele tenha conteúdo similar aos arquivos do diretório **/**[**ta-gui**](https://github.com/pauloborba/teachingassistant/tree/140d028f9ac3c0bf0e1c44d09828fe4e52b842ec/gui/ta-gui)**/**[**src**](https://github.com/pauloborba/teachingassistant/tree/140d028f9ac3c0bf0e1c44d09828fe4e52b842ec/gui/ta-gui/src). Foque apenas em criar ou modificar os arquivos de módulo, componente, HTML, e CSS. Para criar arquivos em [plnkr.co](https://plnkr.co), digite "src/" antes do nome do arquivo, para que ele seja criado no mesmo diretório que os arquivos existentes. Além disso, nos **imports**, use "src/" ao invés dos "./" que temos nos arquivos do projeto criados localmente. Altere o index.html para referenciar o componente via "app-root", não "my-app". Em caso de problema, analise o **console Javascript** (normalmente uma opção de desenvolvedor) do seu *browser*.
3. Para facilitar a navegação de código armazenado no GitHub, você pode querer instalar o seguinte plugin [Octotree](https://chrome.google.com/webstore/detail/octotree/bkhaagjahfmjljalopjnoealnfndnagc)

## Aula 2

1. Coloque de novo o sistema no ar, como feito na aula anterior e visualize o sistema funcionando no seu *browser*. O código fonte do **sistema (aplicação)** inicialmente criado está em[**ta-gui**](https://github.com/pauloborba/teachingassistant/tree/140d028f9ac3c0bf0e1c44d09828fe4e52b842ec/gui/ta-gui)**/**[**src**](https://github.com/pauloborba/teachingassistant/tree/140d028f9ac3c0bf0e1c44d09828fe4e52b842ec/gui/ta-gui/src)**/app**,consistindo de um único **módulo** Angular contendo um único **componente**. Navegue para esse diretório e cole a seguir a imagem do seu terminal com o conteúdo desse diretório. Que arquivos desse diretório representam o componente? Que arquivos desse diretório representam o módulo?
2. As informações visualizadas no *browser* são definidas pelo código HTML do arquivo **app.component.html.** Abra e analise esse arquivo, comparando com a imagem da questão anterior; por exemplo, busque no arquivo por algumas das palavras que você visualiza no *browser*. Consulte o [manual de referência HTML](https://www.w3schools.com/tags/) (ou a [*cheatsheet*](https://htmlcheatsheet.com)). Dê uma olhada geral nas ***tags*** disponíveis, e mantenha esse *link* entre os seus preferidos, ou use outra forma de rapidamente acessá-lo sempre que necessário. Como é difícil dominar rapidamente todos os detalhes de uma linguagem como essa, e como a linguagem evolui com uma certa frequência, uma importante habilidade de um Engenheiro de Software é consultar rapidamente a documentação de linguagens e APIs. Qual a função da *tag* **h2**? E da *tag* **p**? Há algumas dessas no arquivo **app.component.html**? Quais os efeitos exercidos pelas mesmas na visualização desse arquivo? Qual a função da *tag* **a**? Qual a função do **atributo href** associado à *tag* **a**? Qual a função do atributo **target** associado à *tag* **a**? Que efeito teria tirar o elemento **div** (*tags* **<div>** e **</div>**, não o conteúdo entre elas) do arquivo em questão? Qual a diferença dos papéis exercidos por *tags* e atributos?
3. Para entender melhor a relação entre o que é visualizado em uma página aberta no seu browser e o código fonte HTML responsável por gerar tal visualização, é importante acessar o inspetor *web* do seu *browser*. No Chrome, por exemplo, tente **Visualizar > Desenvolvedor > Inspecionar elementos**. No Safari, tente **Develop > Show Web Inspector**. Cole abaixo uma imagem do seu *browser* na página do sistema, com o inspetor ativado. Explique a seguir como o inspetor representa os elementos do documento HTML sendo apresentado pelo *browser*.
4. Note que a palavra "ta-gui", que aparece logo no topo da página principal do sistema, não aparece no arquivo HTML do componente. No HTML, temos apenas **{{ title }}**, que é responsável pelo ***binding*** (ligação) da informação armazenada no estado do componente para o conteúdo da página. Quando executamos o sistema, a informação **title** do estado do componente é lida e utilizada para preencher o conteúdo da página. Em que arquivo desse diretório é declarada essa parte do estado?
5. Abra agora e analise o arquivo **app.component.ts.** Indique abaixo como ele referencia os arquivos HTML e CSS relacionados a esse componente da GUI. Observe a inicialização da variável **title**. Qual o efeito de modificar o valor dessa inicialização? Consulte o [manual de referência de Angular](https://angular.io/guide/displaying-data), em particular essa parte que explica como dados do estado do componente podem ser referenciados no HTML. Mas guarde a referência para fácil acesso durante todo o curso. Modifique o valor armazenado no estado do componente e observe o efeito na execução do sistema. Após gravar a modificação feita no arquivo, o processo inicializado com o **npm start** invocadoanteriormentedeverá recompilar todo o sistema e já colocar uma nova versão no ar. Se isso não ocorrer, invoque em [**ta-gui**](https://github.com/pauloborba/teachingassistant/tree/140d028f9ac3c0bf0e1c44d09828fe4e52b842ec/gui/ta-gui) o comando **npm install**, que baixa e instala as dependências necessárias para a compilação e execução do projeto (segundo especificado no arquivo **package.json**), e depois invoque o comando **tsc**, quecompila todos os arquivos Typescript (segundo as configurações nos arquivos **tsconfig\***). Cole a seguir a imagem do seu *browser* mostrando a página do sistema após a modificação do estado.
6. Analise os demais arquivos em[**ta-gui**](https://github.com/pauloborba/teachingassistant/tree/140d028f9ac3c0bf0e1c44d09828fe4e52b842ec/gui/ta-gui)**/**[**src**](https://github.com/pauloborba/teachingassistant/tree/140d028f9ac3c0bf0e1c44d09828fe4e52b842ec/gui/ta-gui/src)**/app**,com exceção do arquivo com terminação **spec.ts.** Note as referências e dependências entre os arquivos. Como o arquivo que define o módulo referencia o arquivo que define o componente? Como o arquivo **index.htm**l em[**ta-gui**](https://github.com/pauloborba/teachingassistant/tree/140d028f9ac3c0bf0e1c44d09828fe4e52b842ec/gui/ta-gui)**/**[**src**](https://github.com/pauloborba/teachingassistant/tree/140d028f9ac3c0bf0e1c44d09828fe4e52b842ec/gui/ta-gui/src), a página HTML inicialmente mostrada pelo *browser* quando o sistema é invocado, referencia o componente definido?
7. Verifique o status do seu repositório e que não há modificação a ser gravada. Em um novo *branch*, mude o seu sistema para que ele tenha as funcionalidades presentes no *commit* **elementos do formulario para cadastro de alunos** (**não no *master*, mas no *branch* criado pelo professor, disponibilizado no GitHub, e associado à essa edição da disciplina, por exemplo, 2019-2, 2020-1, etc.**). A melhor forma de fazer isso é digitando, enquanto no novo *branch* criado, **git cherry-pick hash-do-commit-mencionado** (e fazendo o mesmo para *commits* anteriores a ele que ainda não foram trazidos para o novo *branch*). Resolva os conflitos, se houver algum, e conclua o processo com **git cherry-pick --continue**. Execute a nova versão do sistema com as mudanças trazidas pelo **cherry-pick**. Não havendo problemas, integre a mudança do novo *branch* ao seu *master*, e cole logo abaixo a imagem da nova página do sistema em execução.
8. Analise os arquivosfontesdessa nova versão do sistema. Observe a classe auxiliar definida no arquivo Typescript do componente. Consulte o manual de referência de [Typescript](https://www.typescriptlang.org/docs/home.html) e [Javascript](https://www.w3schools.com/jsref/default.asp), e esteja pronto para acessá-los muitas vezes durante o curso. Tire dúvidas com a equipe de ensino sobre termos que aparecem nesses manuais e que atrapalham a sua leitura dos mesmos. O que representa a notação abaixo nessas linguagens?

**{nome: "", cpf: "", email: ""}**

1. O que acontece quando você remove um elemento **label** do HTML do componente? E um **input**? Qual o efeito da declaração **[(ngModel)]="aluno.cpf"?** Qual o efeito de modificar a inicialização da variável **aluno** no componente? Faça essa modificação e cole a seguir a imagem da nova página do sistema após essa modificação. Note que o **[(ngModel)]** estabelece um *binding* bidirecional. Qual a diferença do mesmo para a notação usada em **{{ title }}?** Consulte o [manual de referência de Angular](https://angular.io/guide/displaying-data).
2. Modifique o seu projeto para que ele agora considere uma informação a mais sobre os alunos: o *login* GitHub de cada aluno. Cole abaixo a imagem da nova página do sistema em execução. Se tudo estiver funcionando, pode dar *commit*.
3. Modifique o seu sistema, em cima do que foi feito na questão anterior, para que ele tenha as funcionalidades presentes no *commit***botao e funcionalidade do formulario para cadastro de alunos***.* Siga as recomendações dadas anteriormente para copiar essas mudanças a partir do ***branch* criado pelo professor e associado à essa edição da disciplina (por exemplo, 2019-2, etc. na dúvida, pergunte ao professor que branch usar)**. A partir de agora, nas questões seguintes, deixaremos essa recomendação implícita. Observe a mudança no arquivo HTML. Qual o efeito da mesma? Observe onde o método **gravar** foi definido, e o seu **comportamento**. Note que a nova classe de serviço está realizando o serviço que deveria ser feito pelo servidor. Como estamos focando primeiro na GUI, e ainda não temos código do servidor, essa classe de serviço atualmente é um ***stub***. A sua versão final será responsável por enviar requisições HTTP ao servidor, que então será responsável por armazenar os alunos. Por que você acha que a classe **Aluno** foi movida do arquivo do componente para um novo arquivo? Execute e teste o sistema, e cole logo a seguir a imagem da nova página do sistema em execução.
4. Modifique o seu sistema para que ele tenha as funcionalidades presentes no *commit* **visualizacao da lista de alunos cadastrados**. Analise os arquivosfontesmodificados nessa nova versão do sistema. Qual a vantagem do componente manter seu próprio *array* de alunos, em adição ao armazenado na classe de serviço (assumindo que esta representa informações que estarão no servidor)? Qual a semântica da **diretiva \*ngFor**? O que aconteceria se as chaves não fossem usadas em **{{a.nome}}**? Observe como os commits contêm um conjunto reduzido de mudanças, com um foco específico, e com uma mensagem que descreve claramente esse foco. Qual a vantagem dessa prática? Execute e teste o sistema, e cole logo a seguir a imagem da nova página do sistema em execução, depois de você ter cadastrado pelo menos dois alunos.
5. Modifique o seu sistema para que ele tenha as funcionalidades presentes no *commit* **melhor visualizacao do botao e da lista de alunos**. Analise os arquivosfontesmodificados nessa nova versão do sistema. Consulte o [manual de referência de CSS](https://www.w3schools.com/css/default.asp) e o [mapa de código de cores](https://www.w3schools.com/colors/colors_picker.asp). Qual o efeito do atributo **class** no arquivo HTML? Qual o efeito do **button:hover** no arquivo CSS? Execute e teste o sistema, e cole logo a seguir a imagem da nova página do sistema em execução, depois de você ter cadastrado pelo menos dois alunos.

## Aula 3

1. Verifique como depurar o código do cliente com uma ferramenta de depuração (como a [provida](https://www.digitalocean.com/community/tutorials/how-to-debug-angular-cli-applications-in-visual-studio-code) pelo VSCode, por exemplo) ou a de inspeção do *browser***.**
2. Modifique o seu sistema para que ele tenha as funcionalidades presentes no *commit* **evitar cadastro de mais de um aluno com o mesmo cpf**. Analise os arquivosfontesmodificados nessa nova versão do sistema. Qual a **semântica** (comportamento) do método **find**? A **sintaxe** **=>** é usada para definir o que? O que aparece antes dessa sintaxe? E após? Por que a verificação (de existência de aluno com mesmo CPF) foi tanto feita no componente quanto na classe de serviço? No método **gravar** de **AppComponent**, Por que só o atributo **cpf** éatualizado com a *string* vazia quando tenta-se cadastrar um aluno com CPF já existente? Execute e teste o sistema, e cole logo a seguir a imagem da nova página do sistema em execução, depois de você ter cadastrado pelo menos dois alunos e tentando cadastrar um com CPF já existente.
3. Modifique o seu sistema para que ele tenha as funcionalidades presentes no *commit* **refatoracao do metodo gravar**. Analise os arquivosfontesmodificados nessa nova versão do sistema. Por que você acha que o desenvolvedor resolveu **refatorar** o método **gravar**? Execute e teste o sistema. Ele tem o mesmo comportamento que a versão anterior do sistema? Por que?
4. Modifique o seu sistema para que ele tenha as funcionalidades presentes no *commit* **alerta para CPF já cadastrado**. Analise os arquivosfontesmodificados nessa nova versão do sistema. Qual o comportamento do comando **alert**? Execute e teste o sistema, e cole logo a seguir a imagem da nova página do sistema em execução, depois de você ter cadastrado pelo menos dois alunos e tentando cadastrar um com CPF já existente.
5. Modifique o seu sistema para que ele tenha as funcionalidades presentes no *commit* **mensagem, ao invés de alerta, para CPF já cadastrado**. Analise os arquivosfontesmodificados nessa nova versão do sistema. Qual a semântica da diretiva **ngIf**? Qual o papel do atributo **cpfduplicado** no estado do componente? Explique a seguir a lógica utilizada para o **tratamento do erro** de cadastro de aluno com CPF existente. Sem o *binding* de **mousemove**, qual seria a diferença no comportamento do sistema? Execute e teste o sistema, e cole logo a seguir a imagem da nova página do sistema em execução, depois de você ter cadastrado pelo menos dois alunos e tentando cadastrar um com CPF já existente.
6. Modifique o seu sistema para que ele tenha as funcionalidades presentes no *commit* **serviço de aluno injetado, não criado pelo componente**. Ele faz com que uma instância do serviço seja implicitamente criada pelo ***framework,*** e então usada para inicializar o atributo **alunoService** do componente. Analise os arquivosfontesmodificados nessa nova versão do sistema. Por que isso é interessante quando vários componentes forem usar esse serviço, em versões estendidas desse sistema? Você acha que, do commit anterior para esse, houve mudança no **comportamento observável** (que pode ser notado por usuários do sistema) do sistema? Por que?
7. Modifique o seu sistema para que ele tenha as funcionalidades presentes no *commit* **cadastro de metas, e clonagem de objetos…**. Analise os arquivosfontesmodificados nessa nova versão do sistema. Note que o arquivo **Aluno** agora armazena também as metas e os conceitos associados, através de um mapeamento do nome da meta (Especificar requisitos com qualidade, por exemplo) para o conceito associado (MA, por exemplo). Considere agora o arquivo HTML, e observe o tratamento diferenciado para as três primeiras colunas da tabela. Qual a semântica do código HTML associado à quinta coluna da tabela? Por que você acha que, no arquivo do componente, o desenvolvedor preferiu as inicializações de alunos via chamadas ao construtor da classe? Por que você acha que ele renomeou o método **gravar**? Por que você acha que é criado um *clone* do aluno antes de cadastrá-lo na classe de serviço? Execute e teste o sistema, e cole logo a seguir a imagem da nova página do sistema em execução, depois de você ter cadastrado alguns alunos e conceitos de metas.
8. Modifique o seu sistema para que ele tenha as funcionalidades presentes no *commit* **modularizando o componente principal**. Note como o HTML foi melhor modularizado, e quebrado em partes menores que agora fazem partes de dois novos componentes. Como você identifica esses componentes no código fonte do sistema? Qual o papel de cada um dos novos componentes na nova organização do código do sistema? Qual a semântica do atributo **routerLink** no componente principal? Que componente é referenciado pelo *link* "**/metas**"? Onde essa associação é definida? Qual o comportamento do componente **Aluno** emtermos do que ele apresentapara o usuário? E o componente **Metas**? Qual o comportamento da classe **meta** no HTML desse último componente? E do *binding* de **change**? Qual a função de cada um desses bindings usados nos *inputs* desse HTML? Execute e teste o sistema, e cole logo a seguir a imagem da nova página do sistema em execução, depois de você ter cadastrado alguns alunos e conceitos de metas.
9. Opcionalmente, siga o [tutorial](https://angular.io/tutorial) de Angular. Guarde a URL para consultas sempre que encontrar um problema ou precisar de explicações adicionais.