Aplicações que fornecem serviços pela Internet são cada vez mais comuns.

Muitas vezes são serviços simples, com forte apelo comercial e não exigem conhecimento

técnico avançado. Entretanto, aplicações Web também são usadas para grandes sistemas,

principalmente em ambientes corporativos.

As tecnologias de desenvolvimento Web evoluíram bastante desde o início das

primeiras aplicações dinâmicas. De tempos em tempos surgem tecnologias que prometem

ainda mais facilidade no desenvolvimento e resultados mais rápidos. Entretanto, quando se é

preciso desenvolver sistemas de grande porte, com restrições de qualidade rígidas, regras de

negócio complexas e um longo período de evolução e manutenção, a escolha da plataforma de

desenvolvimento geralmente é feita pelas tecnologias mais maduras e com maiores recursos.

Plataformas mais consagradas possuem um número maior de colaboradores,

reunidos como grupos de desenvolvedores ou de empresas que se beneficiam com o sucesso

da tecnologia. Isso conseqüentemente reflete na quantidade de recursos disponíveis,

distribuídos de forma gratuita ou comercial. Esses recursos podem ser encontrados na forma

de ferramentas, documentação, listas de discussão, etc.

O desenvolvedor de sistemas corporativos dessas plataformas se aproveita

bastante de todos esses recursos, principalmente das ferramentas e das “soluções já prontas”

que são uma ótima forma de evitar “reinventar a roda” (FOWLER, 2003). É inevitável na

elaboração de arquiteturas de grandes sistemas ser necessário recorrer a soluções já

implementadas de problemas comuns, principalmente nas camadas de mais baixo nível.

Uma das formas para reutilizar soluções para problemas comuns é a utilização

de *frameworks*. *Framework* de aplicação é um conjunto de artefatos que se complementam e

solucionam um problema comum de maneira genérica (PRESSMAN, 2006). Ele deve

disponibilizar pontos de configuração que possibilitem a customização do comportamento de

seus recursos. Os artefatos normalmente fornecidos são algumas bibliotecas com a

implementação das funcionalidades básicas, modelos de arquivos de configuração para a

personalização dos seus serviços e a documentação para guiar os desenvolvedores no uso da

tecnologia.

2

Em sistemas de grande porte, diversos *frameworks* podem compor uma

arquitetura sendo responsáveis pelos serviços básicos de muita importância como persistência

de dados, controle de transacionalidade das operações, controle de processamento distribuído

e criação de componentes de *interface* gráfica. A utilização de soluções “semi-prontas” traz

grandes vantagens para o desenvolvimento desses sistemas, como permitir que os

desenvolvedores possam se concentrar na implementação de componentes de mais alto nível e

diminuir todo o custo que envolve o desenvolvimento da infra-estrutura do sistema.

Existem muitos tipos de *frameworks* de aplicação com objetivos e

funcionalidades distintas. No desenvolvimento de aplicações Web, os *frameworks* que

auxiliam na infra-estrutura responsável pela manipulação das requisições HTTP e a

formatação e geração do conteúdo respondido às máquinas clientes podem ser chamados de

*frameworks* Web. Na plataforma Java existem diversos *frameworks* desse tipo, distribuídos

gratuita e comercialmente.

Atualmente os *frameworks* Apache Struts e JavaServer Faces (JSF) são os dois

*frameworks* Web mais cotados a serem utilizados em projetos orientados a objetos para

grandes sistemas Web baseados na plataforma Java. O Struts, baseado no tratamento de ações

de formulários de páginas HTML, ainda hoje domina o mercado com sua primeira e mais

popular versão (1.2) por ter sido uma das primeiras implementações que atendia aos

principais requisitos para esse tipo de aplicação. Por outro lado o JSF, lançado no fim de

2004, é hoje anunciado como a API oficial da plataforma JavaEE e tem como principais

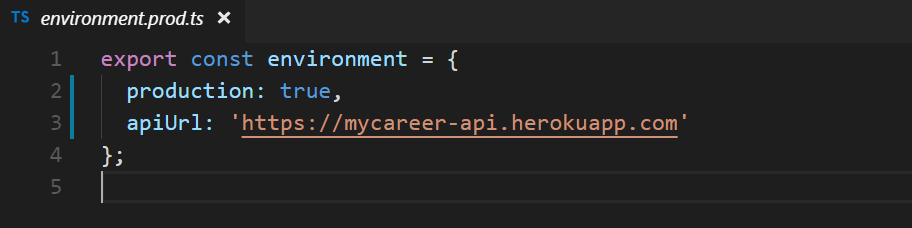
vantagens seu suporte a ferramentas de desenvolvimento RAD e o gerenciamento de eventos

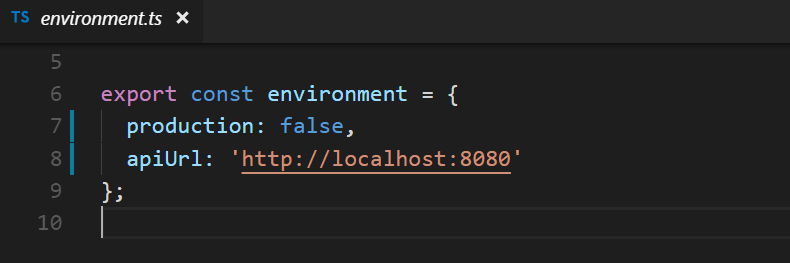
de todos os componentes da *interface* do usuário exibidos do lado cliente da aplicação.

Deploy WebJobs

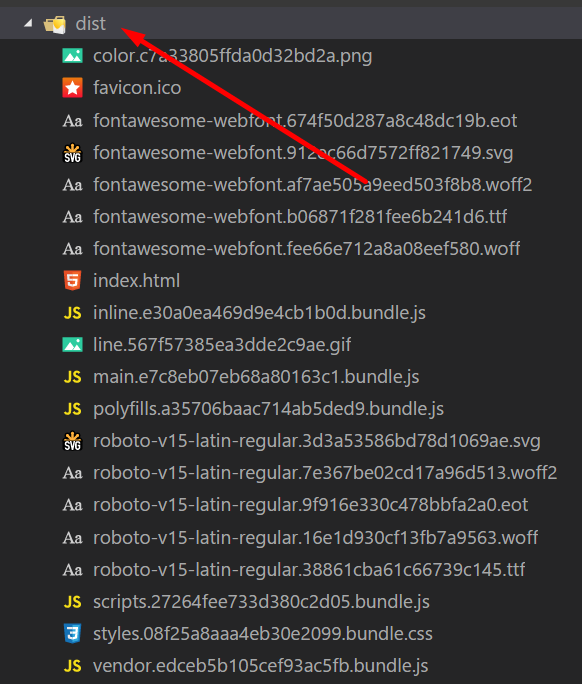
<https://mycareer-webadmin.herokuapp.com/>

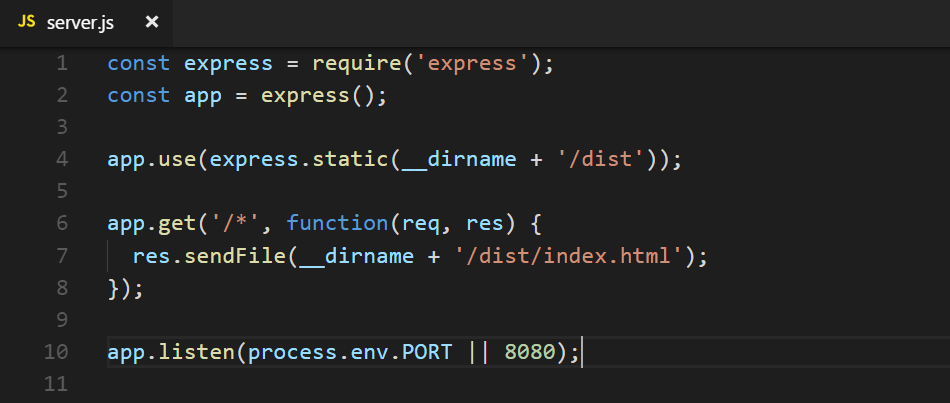
<https://mycareer-webajobs.herokuapp.com/>



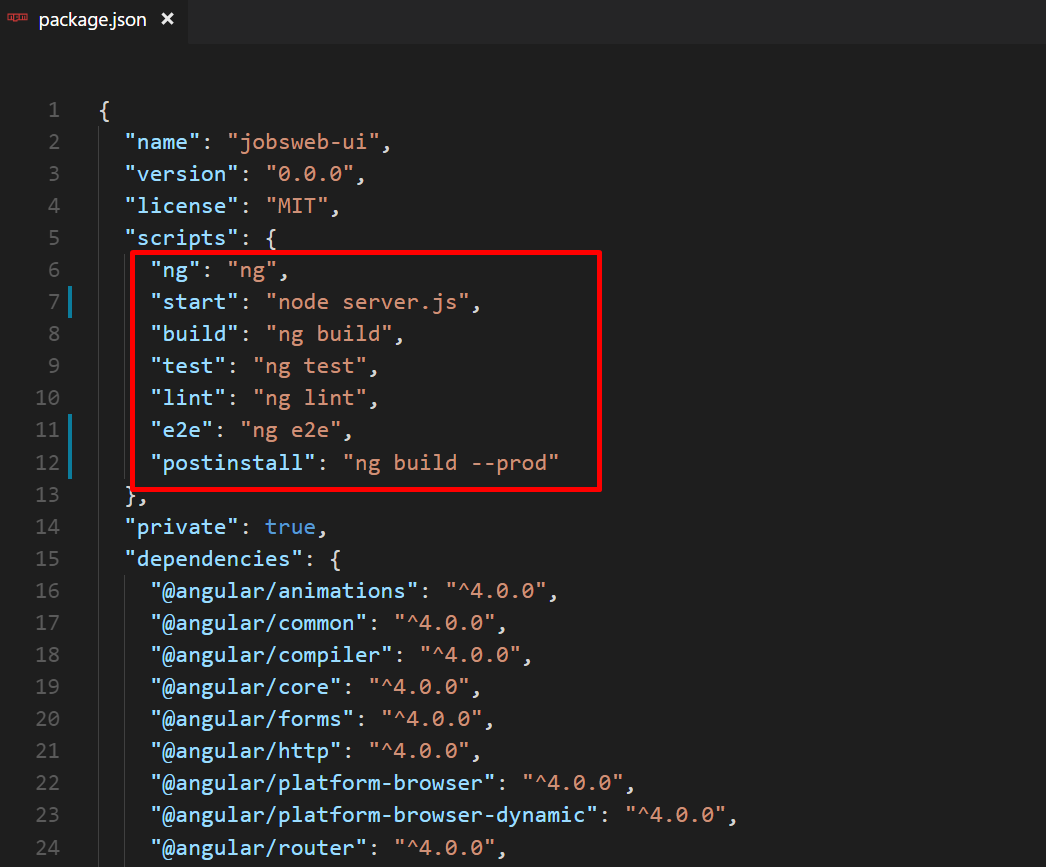


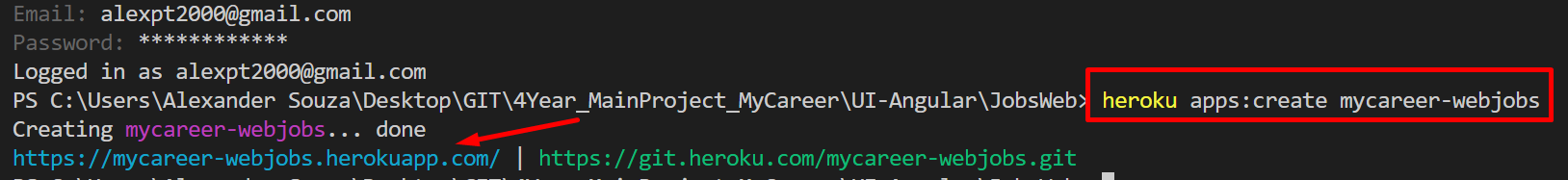




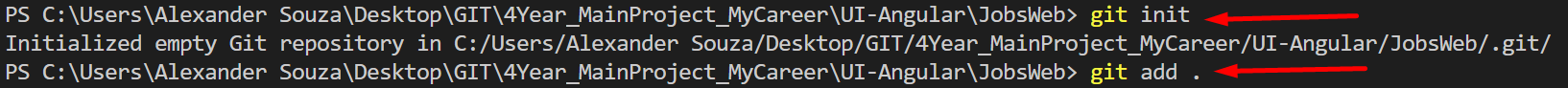


Node server.js



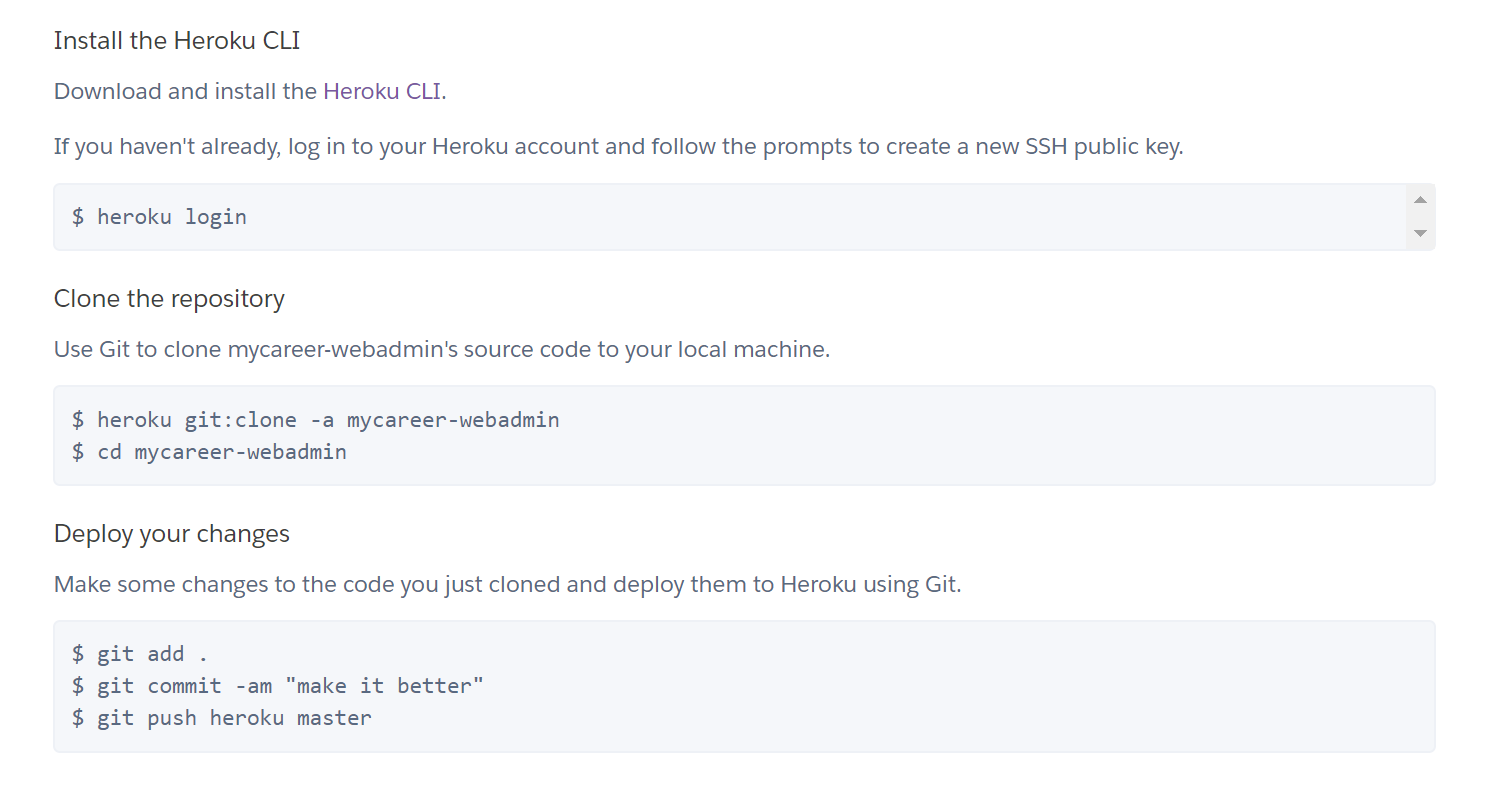


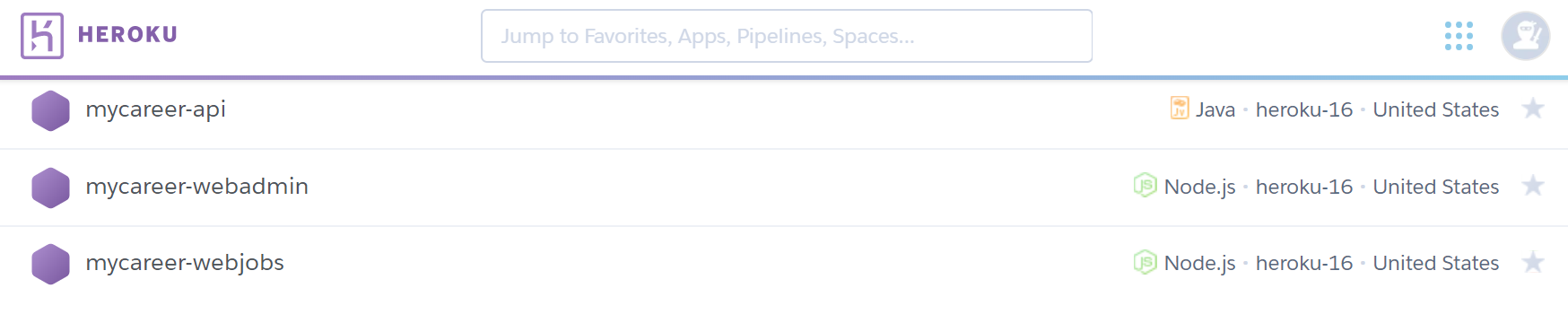












Swagger

<https://mycareer-api.herokuapp.com/swagger-ui.html>

Ref:

<http://www.baeldung.com/swagger-2-documentation-for-spring-rest-api>