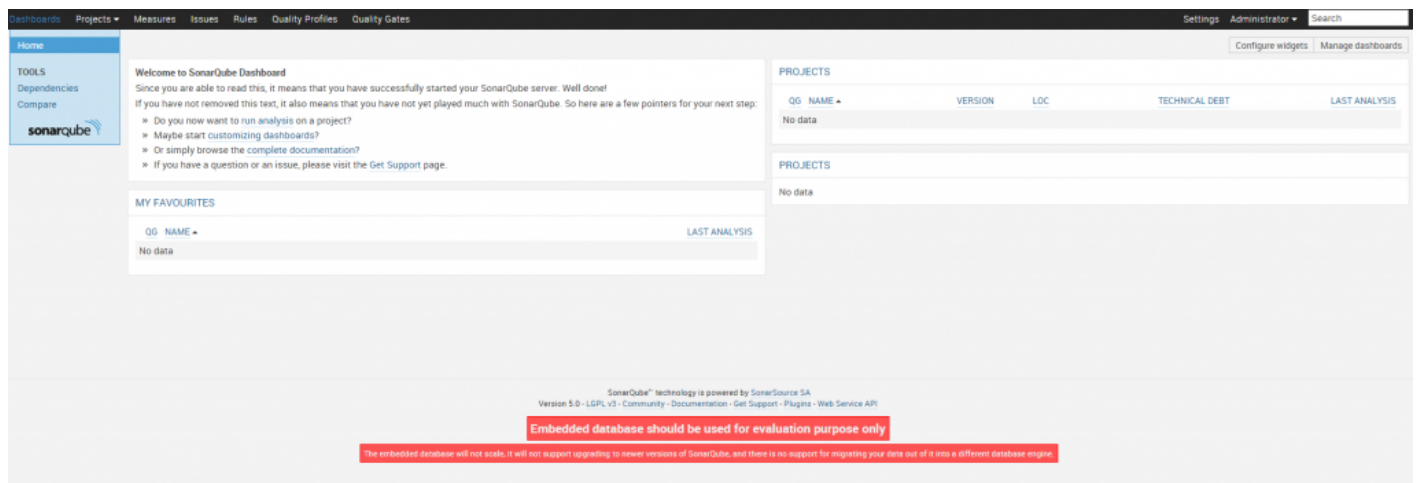


INSTALANDO E CONFIGURANDO LOCALMENTE

1. Em primeiro lugar, baixe o SonarQube (web server) e o SonarQube Runner (ferramenta linha de comando que faz a análise do código). Os links estão [<aqui>](#) (baixe a versão 5.0 do SonarQube).
2. Descompacte os arquivos respectivamente em: “C:\BuildTools\sonarqube” e “C:\BuildTools\sonarqube-runner”.
3. Na pasta “bin” do SonarQube, há uma pasta para cada sistema operacional suportado. Para rodá-lo no Windows (64 bits), vá até a pasta correspondente e execute o **StartSonar.bat**. Após algumas instruções executadas no console, a última linha deverá mostrar “Process[web] is up”. Navegue, então, para o endereço “http://localhost:9000” e você deverá ver a página inicial do SonarQube:



Obs.: É recomendado que o SonarQube rode como um serviço Windows. Isso também é bem simples de ser feito. Veja [<aqui>](#).

4. Reparem no aviso em vermelho na imagem acima. Por padrão, o SonarQube utiliza um embedded database. Para fins de demonstração, como este artigo, não há problemas, mas para colocá-lo em produção, você DEVE configurá-lo para o uso com algum banco de dados “de verdade”. Não entrarei neste tópico mas a configuração também é fácil – basta fazer algumas configurações nos arquivos .properties da pasta “conf” (tanto na pasta do Sonar como na pasta do Runner). Mais detalhes [<aqui>](#).

PLUG-IN DO C#

5. Antes de efetuarmos a análise de um código C# é preciso instalar o plug-in da linguagem. Para isso, faça login no SonarQube, no menu superior direito, usando as credenciais-padrão: **admin**, tanto no login quanto na senha.
6. Após logado, vá para o Update Center em http://localhost:9000/updatecenter e clique em “Available plugins”. Você verá que existem dezenas de plug-ins para diversas linguagens, para integração com outras ferramentas, visualização de relatórios e gráficos, *language packs*, entre outros. Clique em “C#” e, em seguida, no botão “Install”.
7. Para concluir a instalação do plug-in, você terá que reiniciar o SonarQube, fechando o console e reexecutando o .bat ou reiniciando o serviço, caso tenha instalado-o como tal.

8. Retornando ao Update Center, você deverá ver o plug-in do C# já instalado na primeira aba (Installed Plugins). Agora o SonarQube está apto a analisar código C#, levando em conta **26 regras** de análise. Falaremos dessas regras em breve.

RODANDO A ANÁLISE

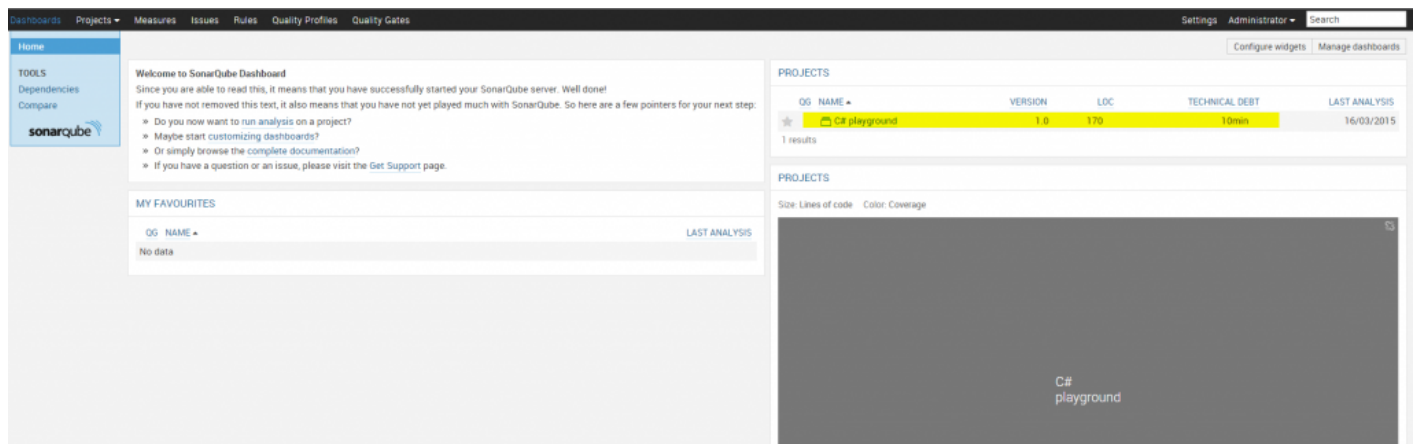
9. Para fazermos um teste, vamos rodar o analisador com um projeto que o pessoal do SonarQube disponibiliza com exemplos em dezenas de linguagens. Faça o download [aqui](#).

10. Descompacte o zip em “C:\sonar-examples\”.

11. Adicione o caminho do Sonar Runner ao path do Windows em variáveis de sistema. Se você usou os mesmos nomes utilizados neste passo-a-passo, o caminho deverá ser: “C:\BuildTools\sonarqube-runner\bin;”.

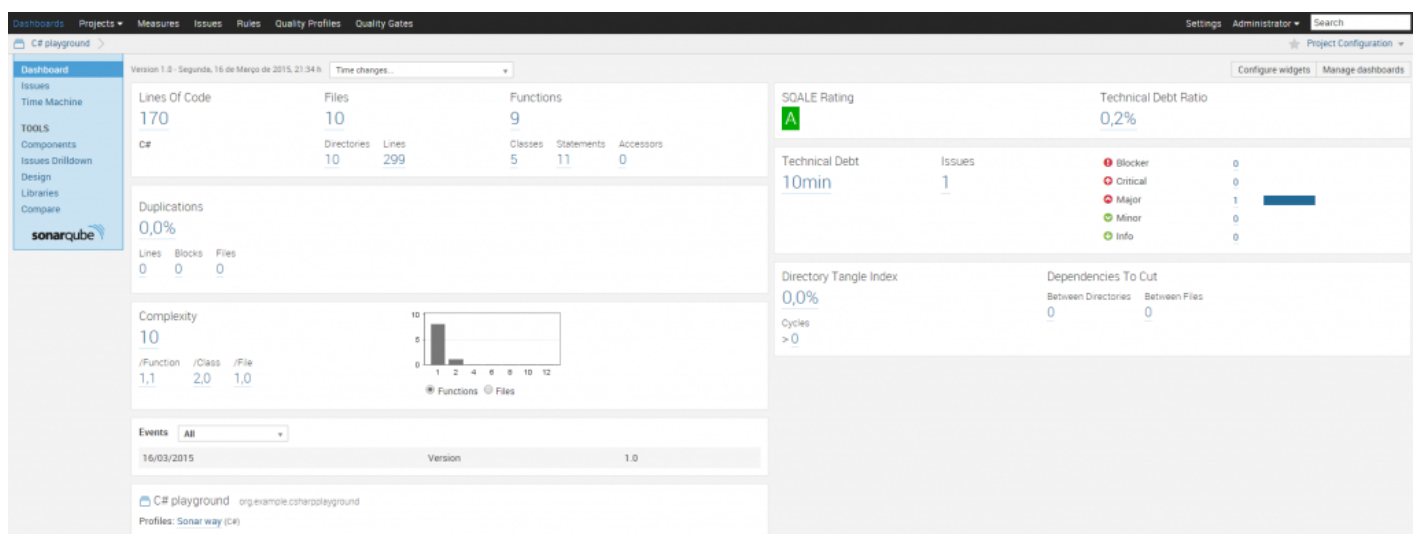
12. Pelo prompt de comando, navegue até a pasta do exemplo em C# (C:\sonar-examples\projects\languages\csharp) e execute o comando “sonar-runner”.

13. Concluída a análise, atualize a página do Sonar e você deverá ver um projeto novo de nome “C# Playground”, marcado em amarelo na imagem:



The screenshot shows the SonarQube dashboard. On the left, there's a sidebar with 'Home', 'TOOLS', and 'sonarqube' logo. The main area has a 'Welcome to SonarQube Dashboard' message. On the right, the 'PROJECTS' table lists one project: 'C# playground' with version 1.0, 170 LOC, 10min technical debt, and last analysis on 16/03/2015. Below the table, there's a 'C# playground' visualization area.

14. Clique no nome do projeto “C# Playground” e você irá para a *dashboard* do mesmo, que mostra uma série de informações coletadas durante a análise:



The screenshot shows the project dashboard for 'C# playground'. It displays various metrics: Lines Of Code (170), Files (10), Functions (9), Duplications (0.0%), Complexity (10), and a bar chart for Complexity. On the right, it shows SQALE Rating (A), Technical Debt Ratio (0.2%), Technical Debt (10min), Issues (1), and a breakdown of issues by severity (Blocker, Critical, Major, Minor, Info). It also shows Directory Tangle Index (0.0%) and Dependencies To Cut (0).

Observem que a *dashboard* fornece várias métricas (lado esquerdo) como número de linhas de código, número de classes, duplicações de código e complexidade ciclomática. No lado direito, são exibidas informações sobre dívida técnica, como por exemplo, a quantidade de problemas (*Issues*), ou seja, o número de regras que foram quebradas.

As *issues* são categorizadas quanto ao seu grau de severidade, indo de “Blocker” (mais grave) até “Info” (mais leve).

(Se quiser ver uma *dashboard* com mais informações, dê uma olhada nesse Sonar online, com a análise de diversos projetos open-source: <http://nemo.sonarqube.org/>.)

Simples, não?

.....

ANALISANDO SEU PRÓPRIO CÓDIGO

Quer analisar um projeto seu? A ideia é a mesma. Execute o comando “sonar-runner” na pasta raiz do projeto, onde se encontra o arquivo .sln.

Antes disso, porém, é preciso que você crie e configure o arquivo **sonar-project.properties** no mesmo local do .sln (perceba que o exemplo acima possuía esse arquivo). Um arquivo .properties mínimo para rodar o projeto poderia ser algo como:

```
sonar.projectKey=IDENTIFICADOR
sonar.projectName=NOME DO PROJETO
sonar.projectVersion=1.0
sonar.sourceEncoding=UTF-8
sonar.sources=.
sonar.language=cs
```

Altere IDENTIFICADOR e NOME DO PROJETO com suas preferências.

Vale ressaltar que este arquivo pode (e certamente irá) conter mais uma dezena de configurações, referentes a cobertura de código, exclusões, entre tantas outras.

.....

MAIS ALGUNS DETALHES

- Como dito antes, o plug-in do C# fornece 26 regras de análise. Essas regras ficam agrupadas em perfis (Quality Profiles) e você pode ativar/desativar regras, assim como criar novos perfis.
- Vinte e seis regras é um número pequeno para uma análise mais efetiva. Por isso, utilizamos, em conjunto, o **plug-in do Resharper**, instalado também pelo Update Center do SonarQube. Com isso, você ganhará todas as inspeções do Resharper, ou seja, mais de 600 regras novas! Saiba mais [<aqui>](#).

- É possível também utilizar as regras do Code Analysis do Visual Studio (aka, FxCop).
- O SonarQube se integra a vários servidores de integração contínua. Aqui na empresa, usamos o **Jenkins** para executar a análise do SonarQube, como um job da pipeline de build. Esta é uma prática recomendadíssima, pois a análise será sempre disparada automaticamente e feita sobre o código integrado mais atual no repositório de código-fonte.