Instituto Federal do Espírito Santo

Projeto Capoeira

Briane Bianca

Gustavo Grimaldi Capello

Leticia Schulthais Senna

**Sumário**

1 HISTÓRICO DO DOCUMENTO 2

2 INTRODUÇÃO 3

3 APRESENTAÇÃO 4

3.1 MINIMUNDO 4

3.2 TIPO DE USUÁRIO 4

3.3 DIAGRAMA DE CLASES 5

3.4 DESCRIÇÃO MVC 9

3.5 DESCRIÇÃO PADRÕES DE PROJETO 10

3.6 REFATORAÇÃO E AVALIAÇÃO DE QUALIDADE DE CÓDIGO (SONAR) 11

# HISTÓRICO DO DOCUMENTO

|  |  |  |  |
| --- | --- | --- | --- |
| Versão | Responsável | Modificação | Data de Modificação |
| 1.0 | Elton Couto Rizzo | Criação do documento. | 23/06/2015 |
| 2.0 | Leticia Schulthais Senna | Inserção dos dados | 03/10/2015 |
| 3.0 | Briane Bianca, Gustavo Grimaldi e Leticia Senna | Atualização e inserção de dados | 18/10/2015 |

# INTRODUÇÃO

Projeto Capoeira é um sistema para Professores de Capoeira. Cujo visa substituir as inscrições feitas em fichas impressas para cadastro on-line (acabar com fichas impressas, reduzindo volume, e facilitando a busca/armazenamento dos dados). O Projeto Capoeira é um produto administrativo, para monitorar inscrições, relação de alunos, controle de histórico de alunos, e atividades da capoeira. Que propõe dar agilidade ao processo de ingresso no projeto, facilitar o armazenamento das informações referentes ao que compõem o projeto (alunos, eventos e etc).

# APRESENTAÇÃO

O projeto será inicialmente desenvolvido na linguagem Python, conectando-se com o banco de dados SQLite, e usando uma interface gráfica PyQt4.

## MINIMUNDO

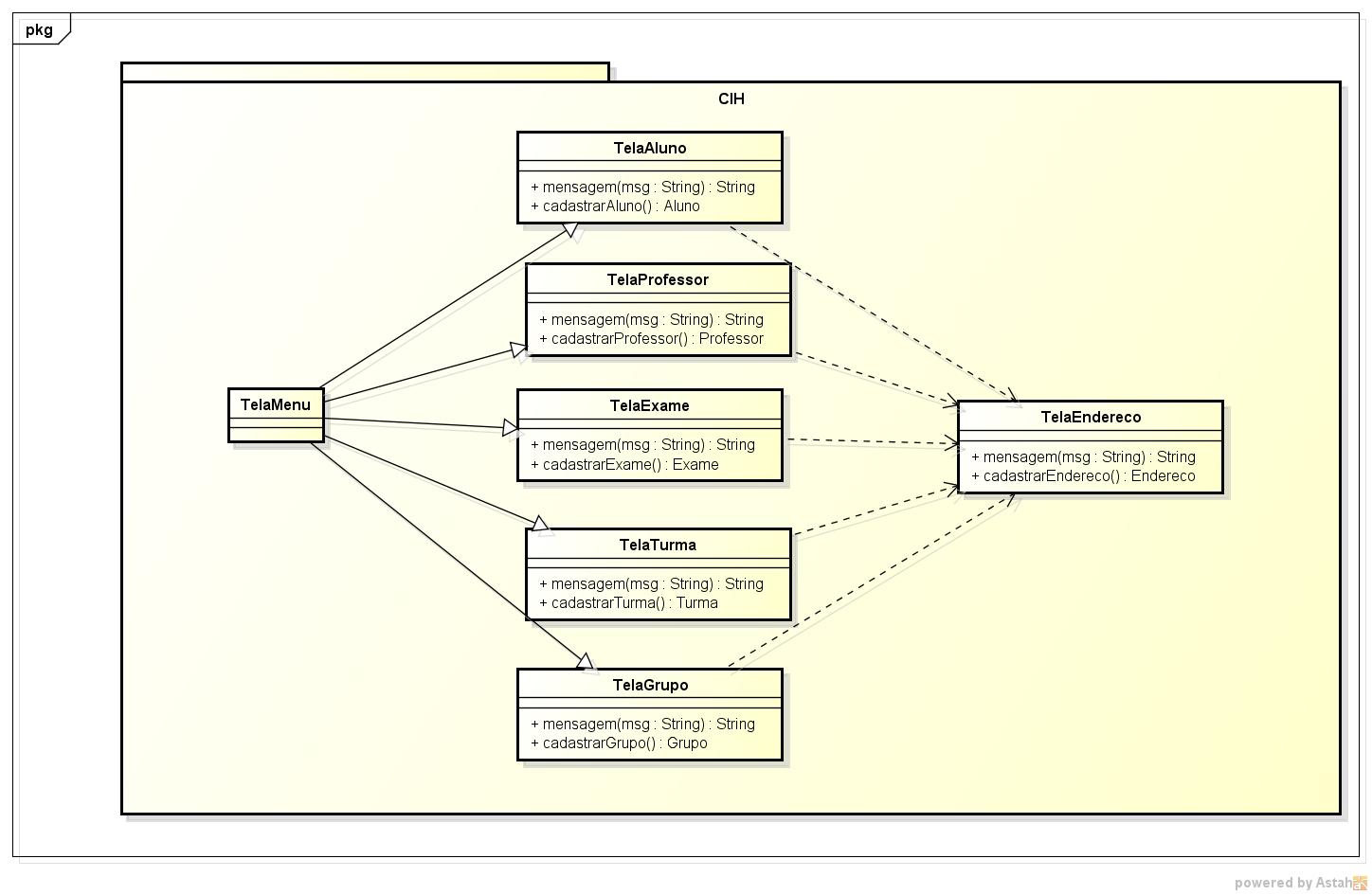
O projeto capoeira é um sistema que armazena digitalmente os dados de alunos, professores, local, entre outras informações relacionadas à prática da capoeira. O sistema deve permitir o cadastro de novos alunos (Nome, RG, Filiação, Data de Nascimento, Endereço, Telefone, Profissão, Grau de Escolaridade, Corda), Professores (Nome, RG, Data de Nascimento, Endereço, Telefone, Profissão, Grau de Escolaridade, Corda), Turmas (Nome, Turno, Horário, Dia da semana), Local (Logradouro, Numero, Bairro, Cidade, Complemento), Corda (Cor), Exame (Data do Exame, Horário, Mestre examinador, Local da Cerimônia, Turma), Grupo (Nome, Endereço, Sequência das cordas). Além disso, o sistema deve gerar um relatório de inscrições ao término de cada de período de inscrição, deve permitir somente um determinado número de inscrições por turma, formando um cadastro de reserva com as inscrições que ultrapassarem o limite.

## TIPO DE USUÁRIO

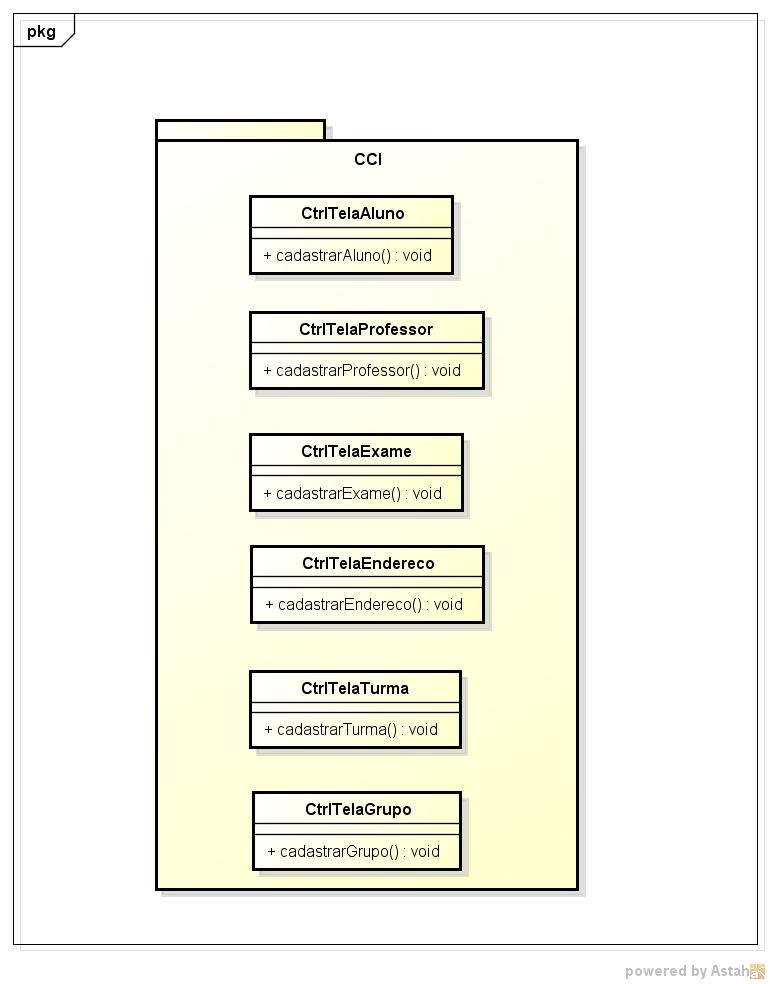
Em geral as pessoas que irão utilizar o sistema possuem um grau de escolaridade suficiente, estando fora de níveis extremos, como por exemplo: o analfabetismo, o que impediria ou dificultaria a utilização do sistema. Deixando claro que o projeto abrange a todos sem qualquer tipo de distinção e que qualquer dificuldade que possa surgir em relação à acessibilidade, seja ela de qualquer forma, será solucionada da maneira mais agradável para ambas as partes.

## DIAGRAMA DE CLASSES

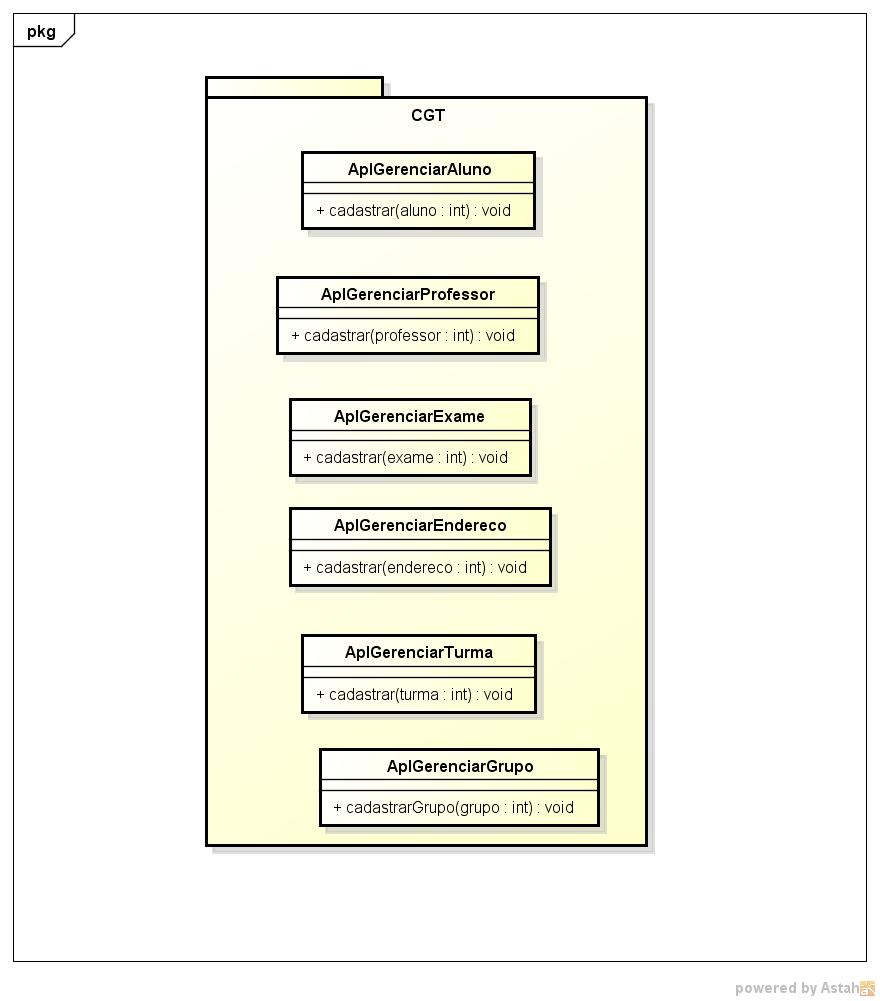
Componente de Interação Humana



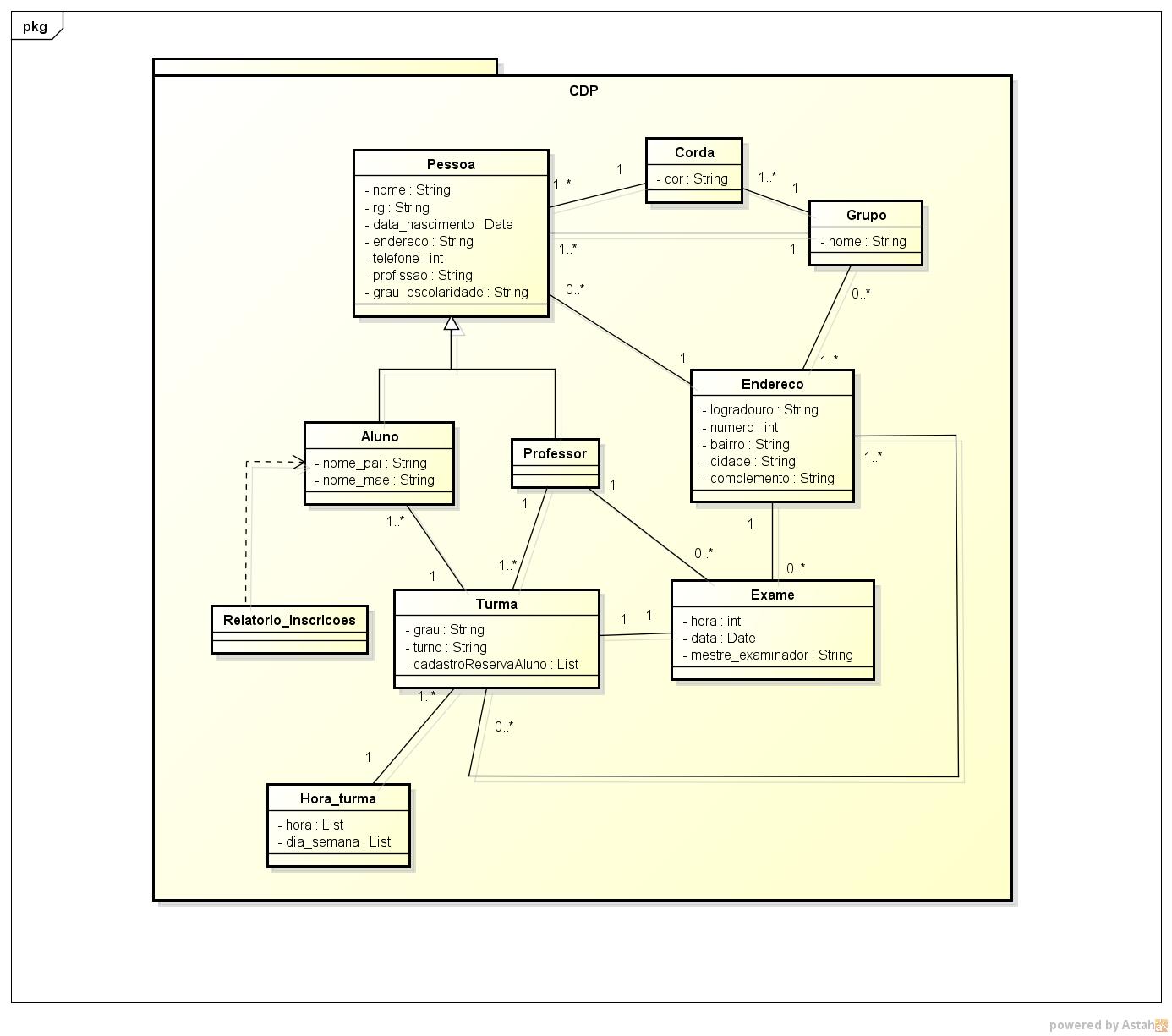
Componente de Controle de Interação



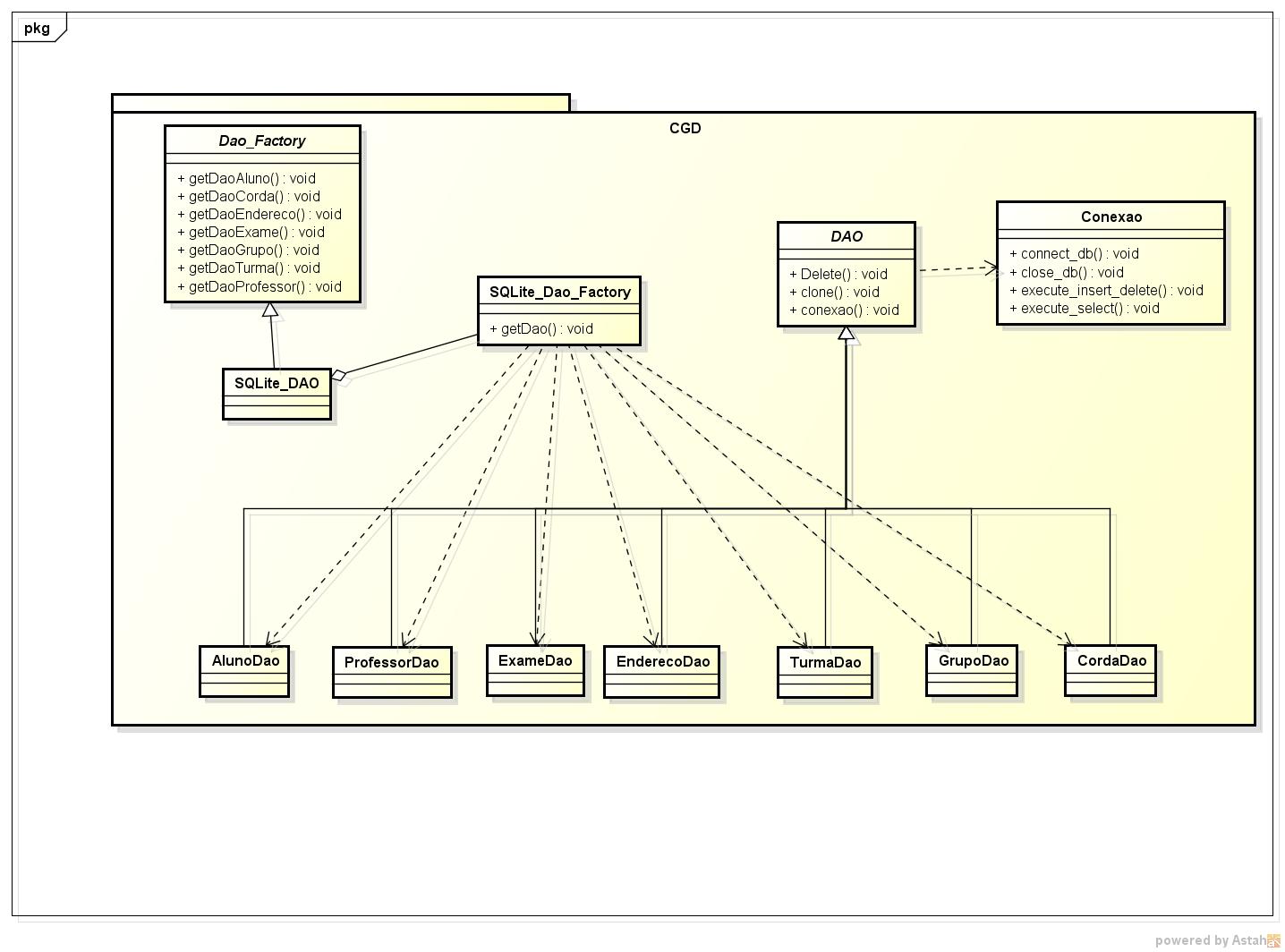
Componente de Gerência de Tarefas



Componente do Domínio do Problema



Componente de Gerência de Dados



## DESCRIÇÃO MVC

* Modelo-visao-controle (MVC – model-view-controller) é um padrão arquitetural de software para implementação da interface do usuário.
* Para cada nível há uma classe responsável por uma ação. No caso deste projeto na camada de visão CIH temos as classes TelaGerir, que fará a chamada no prompt de comando para inserção de dados pelo usuário (com o desenvolvimento deste trabalho faremos a aplicação de uma interface para melhor manuseio do software).
* Na camada de controle CCI estão as classes de CtrlTela, que farão a chamada à classe TelaGerir e receberão os dados inseridos e enviarão para a classe responsável pela persistência dos dados.
* Na camada de modelo estão as classes que criam instanciam os objetos. As classes observadas no modelo acima (seção 3.3), na CGD estão as classes DAO (que realizam as persistência dos dados SQLite), na CGT estão as classes AplGerenciar que recebem os dados da classe CtrlTela e chamam as, e na CDP estão as classes principais do sistema.

## DESCRIÇÃO PADRÕES DE PROJETO

Primeira etapa de padrões de projeto: (Fábrica Abstrata, Método Fábrica, Singleton, Builder e Protótipo)

Utilizamos Fábrica Abstrata, Método Fábrica, Prototipo e Singleton nas classes DAO para a persistência dos dados no banco de dados. O padrão Fábrica Abstrata foi criado para retornar os componentes de persistencia. A classe sqliteDAO contém codigo para manipular o banco SQLite.

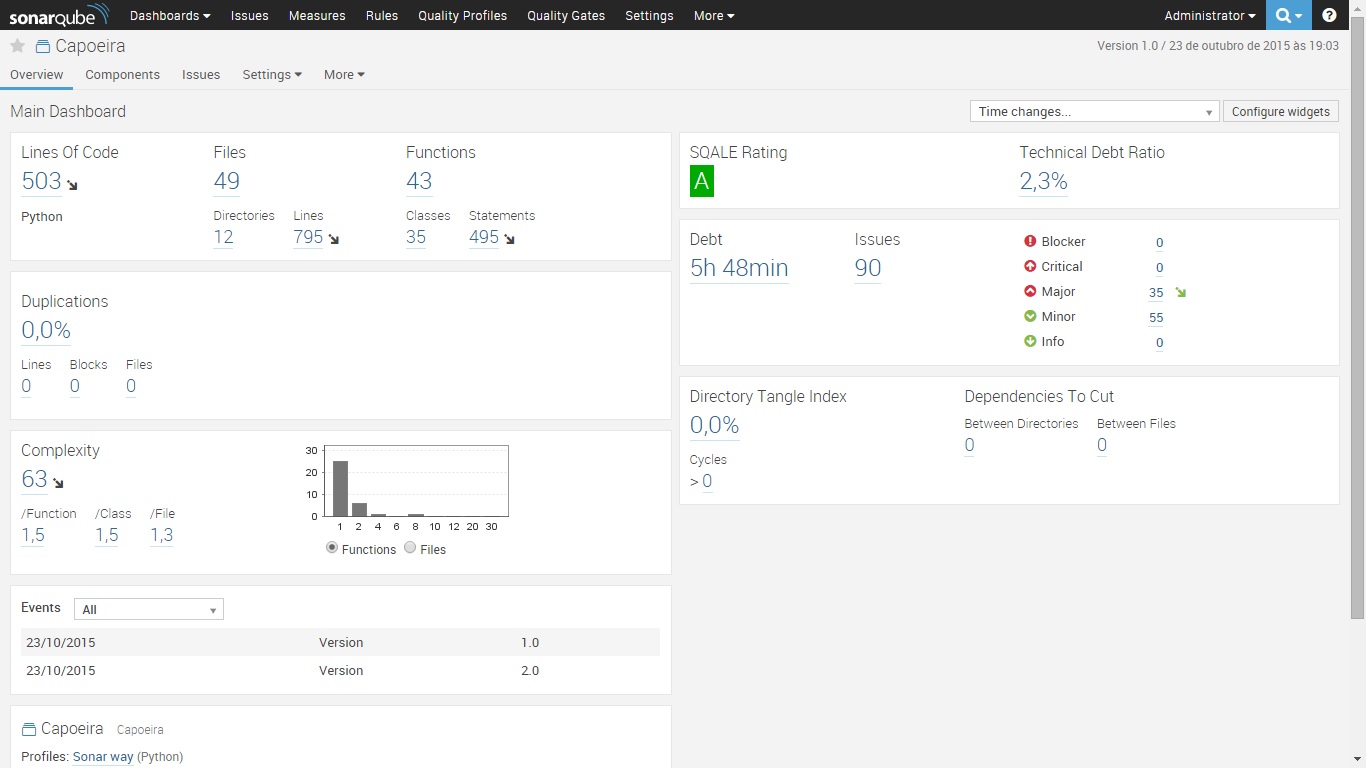
O método Fabrica foi utilizado na classe foi chamada de SQLite\_dao\_.factory para retornar a DAO que fosse solicitada, abstraindo a implementação.

A DAOGeneric que é responsavel por criar todo o banco de dados, realiza conexão e realiza a inserção no banco, possui o padrão Singleton para que a classe seja instanciada apenas uma vez e mantenha a mesma instancia, para a criação de apenas um banco de dados.

Não utilização do padrão Builder: Este padrão tem como objetivo separar a construção de um objeto complexo da sua representação de modo que o mesmo processo de construção possa criar diferentes representações. Deve ser usado em situações onde a construção do objeto deve ser independente das partes que o compõem, de modo que seja possível a existência de diferentes representações para o objeto construído. Visto que em nosso modelo não existe classe que se encaixe nesse padrão, já que todas as classes do modelo tem comportamento semelhante, onde sua construção é dinâmica. Então a maioria das partes, que compõem os objetos, é informada pelo usuário dinamicamente, o que torna inviável a utilização do padrão Builder.

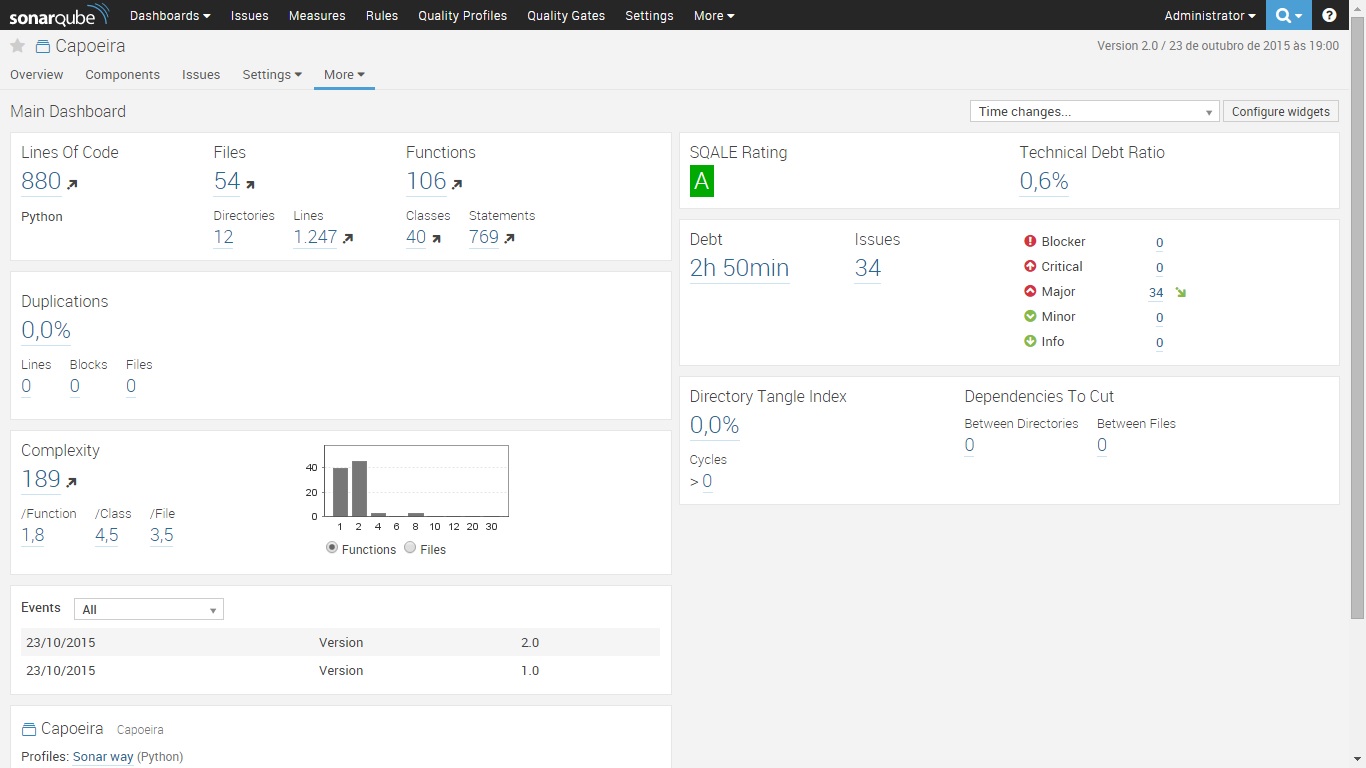
## REFATORAÇÃO E AVALIAÇÃO DE QUALIDADE DE CÓDIGO (SONAR)

* Versão 1:



A primeira versão do projeto, bem simples, sem padrões de projeto.

* Versão 2:



Os padrões ajudaram principalmente na organização do código. Ajudaram a garantir funcionalidades do sistema, buscando eficiência e eficácia. Podemos observar uma redução de débitos técnicos no sistema, o que significa menos instabilidade.

A pesar do Sonar indicar um aumento na complexidade observamos uma redução no tempo do débito e nos problemas. Com a ajuda do mesmo conseguimos reduzir os problemas maiores, zerar os menores e acertar o código onde poderia causar futuros problemas ou erros de execução.

O padrão Singleton melhorou o sistema pelo fato de, além de garantir a integridade do banco de dados, controla a abertura e fechamento do banco. Se não fosse por ele, em casos de processos concorrentes e com duas instâncias da classe conexão, poderia causar problemas na persistência dos dados.

A Fábrica abstraiu o processo de construção dos objetos, trazendo modularidade para o código, pois caso precise ser efetuado alguma modificação futura não impactará muito.

A Fábrica Abstrata serve como molde de fábricas de linguagens diferentes de banco, no caso do projeto foi utilizado apenas o SQLite, mas se quiséssemos adicionar outro banco isso facilitaria.

O Sonar ajudou a visualizar os erros que mais eram de vícios de linguagem, pois no curso programamos muito tempo com Java, e os maiores erros eram nomes de variáveis que usávamos do mesmo jeito que em Java. Os demais erros não puderam ser corrigidos, pois envolviam a criação de métodos abstratos, mesmo modificando a criação desses métodos e adicionando o @abstractmethod o sonar pedia que o método fosse estático.