**Linguagem e tecnologias**

A linguagem utilizada no projeto foi a Java (além de HTML, CSS e JavaScript no front-end). O projeto é um projeto web do Maven. As tecnologias e frameworks envolvidos foram: Maven, CDI, VRaptor, JUnit, JQuery, Bootstrap.

**1) O que vale destacar no código implementado?**

É um código simples, mas que nos revela padrões e boas práticas em desenvolvimento Java. Como foi usado VRaptor como framework MVC, foram exploradas suas funcionalidades e padrões, além dos padrões do CDI usados largamente no VRaptor.

Para controle de acesso foi usado um Interceptor (LoginInterceptor.java), um objeto de sessão (Sessao (@SessionScoped)) e um anotation (@Public (InterceptorBinding)). Todas as classes que tem acesso público (LoginController no caso) deve ser anotadas como ela para que o acesso não seja validado pelo Interceptor. O Interceptor checa se já foi efetuado login, e em caso acertivo dá seguimento a requisição, caso contrário redireciona para a tela de login.

Foram explorados também eventos do CDI. A controller do voto (VotoController) no seu método "save" disparam um evento (Event<Voto>) que por sua vez está sendo "observado" pela classe VotoObserver (método adicionaVoto) demostrando dessa forma prática de baixo acoplamento entre classes.

Também vale destacar a injeção de dependência e o uso de escopos. Além da classe Sessao (@SessionScoped), foi também usado um scopo de aplicação (@ApplicationScoped) para a classe utilitária Repositorio.

Como não foi utilizado banco de dados e sim uma classe utilitária (Repositorio) para simular uma Service/Repository, fui obrigado a fazer um uso mais intenso de Streams/Lambdas do Java 8.

Foram criados testes unitários simples para a classe utilitária e somente para ela, pois é basicamente ela que faz manipulação de dados e os retorna.

**2) O que poderia ser feito para melhorar o sistema?**

Quanto a melhoria, poderíamos fazer inúmeras coisas nesse projeto ainda. Uma delas seria o uso de um banco de dados e com isso a utilização do Hibernate/JPA por exemplo.

Com isso seria necessário implementar um controle transacional.

Teriamos que fazer uso de outros padrões como produtores (@Produces/@Disposes) e escopos para controlar o tempo de vida de objetos como EntityManagerFactory e EntityManager (ou SessionFactory e Session do próprio Hibernate).

No front-end também teria que ser melhorado o visual. Poderíamos também usar templates, ou usar HTML/CSS/JavaScript puros em vez de mesclar com JSTL. AngularJS talvez seria uma outra pedida.

Se estivessemos falando de um projeto grande e multicliente, um SaaS, teriamos que implementar também um esquema de Multi Tenancy (clientes por banco, clientes por schema ou separação por tenant\_id).