DAS (Documento de Arquitetura de Software)

Caça Eventos

Robson de Oliveira Miranda da Silva
Onofre Machado Vieira Neto
Isaias Miranda Leite
Juannyson Ferreira de Carvalho

Teresina – PI Abril de 2018

Sumário

1. Introdução

Uma vez terminada a especificação de requisitos do software, obtém-se a necessidade de se ter uma solução de como deve ser estruturado o software de modo que venha atender as restrições e funcionalidades que o software necessita. E esta solução se definida em partes neste documento de arquitetura de software.

1.1. Finalidade

Este documento apresenta uma visão geral abrangente da arquitetura do projeto Caça Eventos e utiliza uma série de visões arquiteturais diferentes para ilustrar os diversos aspectos do projeto. Ele serve como um meio de comunicação entre o arquiteto de software e outros membros da equipe do projeto com relação a decisões significativas do ponto de vista da arquitetura, tomadas a respeito do projeto.

1.2. Escopo

O Documento da Arquitetura de Software se aplica ao Sistema de Caça Eventos que será desenvolvido pela Integração do Contexto.

1.3. Visão Geral

Após a definição da especificação de requisitos e escopo deste documento, abaixo vamos descrever a arquitetura escolhida que melhor soluciona os problemas e necessidade encontrados neste projeto. Assim vamos apresentar em 5 visões sendo elas casos de uso, lógica, processo, implantação e implementação.

2. Representação da Arquitetura

Este documento apresenta a arquitetura como uma série de visões: visão de casos de uso, visão lógica, visão de processos, visão de implantação e visão de implementação. Essas visões são apresentadas como modelos e utilizam a Linguagem Unificada de Modelagem (UML).

3. Metas e Restrições de Arquitetura

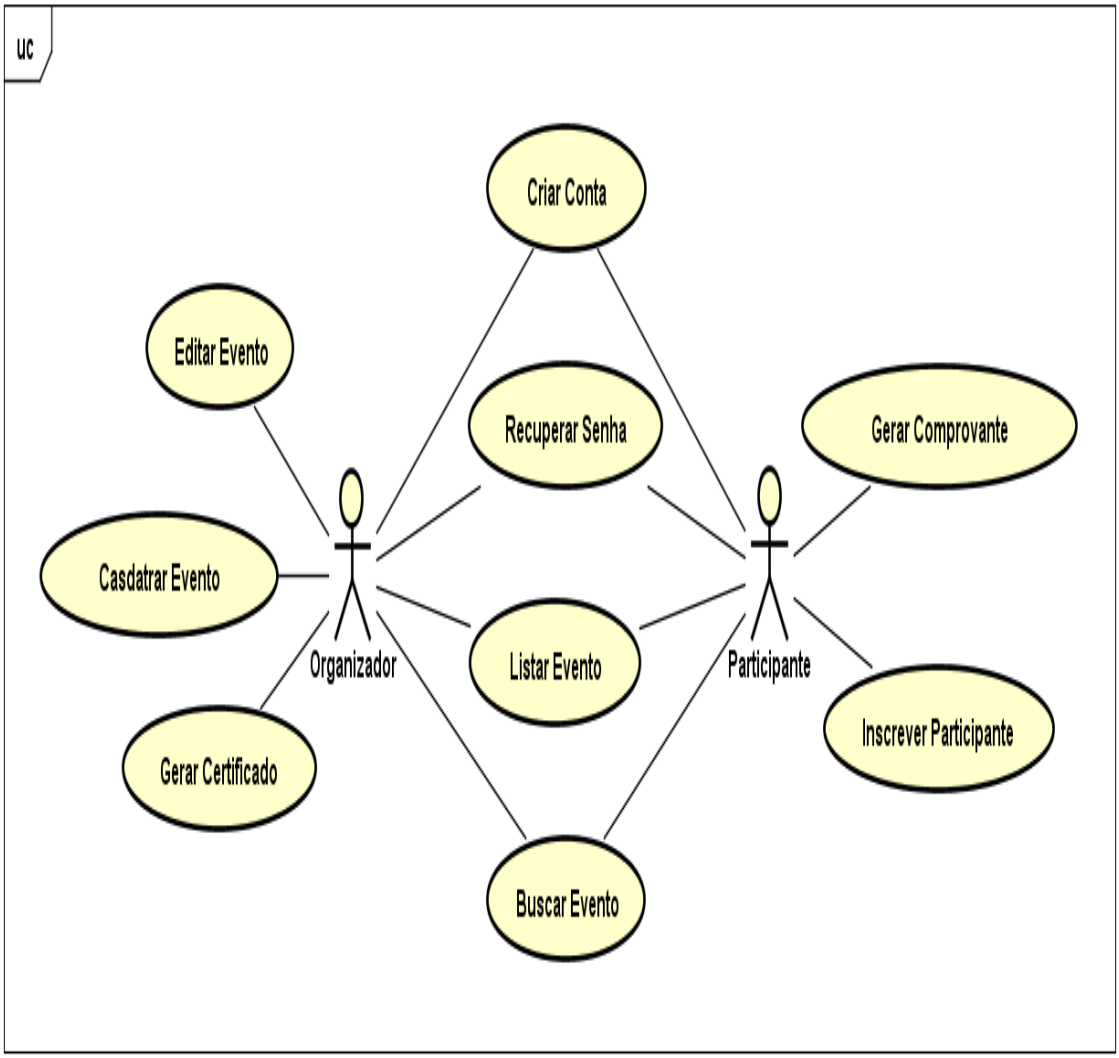
Existem algumas restrições de requisito e de sistema principais que têm uma relação significativa com a arbquitetura. São elas:

Nº	Descrição
1	Autenticação de blockchain na geração de certificado.
2	Ambiente Web, sendo compatível com os principais navegadores do momento: Internet Explorer, Firefox, Safari e Chrome.
3	O servidor deve suportar 100 conexões simultâneas sem perda de desempenho.
4	Acesso restrito por meio de senhas. Controle no registro de senha, de

	forma a impedir o uso de senhas consideradas fáceis.
5	Desenvolvido utilizando a linguagem Python, com a tecnologia Django.
6	Banco de dados SQLite.
7	Interface simples de acesso para o usuário, permitido uma interação humano-computador mais fácil.

4. Visão de Casos de Uso

Os casos de uso nesse sistema estão mostrados abaixo. Uma descrição desses casos de uso pode ser localizada posteriormente nessa seção.



ID	Caso de Uso	Descrição
UC1	Criar Conta	Neste caso de uso, o organizado e o participante inicializa o sistema criando uma conta, onde terão que preenche uma ficha com seus dados

		de cadastro.
UC2	Recuperar Senha	Neste caso de uso, ao esquecer sua senha o usuário seja ele participante ou organizador, solicitar uma nova senha ao enviar o e-mail solicitado.
UC3	Cadastrar Evento	Neste caso de uso, o organizador após Criar a conta no sistema ele irá Cadastrar o Evento que será organizado por ele onde os participantes poderão se inscrever nele.
UC4	Listar Evento	Neste caso de uso, todos os eventos criados pelo organizador e inscritos pelos participantes terá uma lista de cada um deles para obter informações especificas como, por exemplo, localização ou contato do organizador.
UC5	Buscar Evento	Neste caso de uso, qualquer usuário seja organizador ou participante, podem faze busca do Evento de sua preferencia por nome, tipo de evento e estado, mostrando o evento buscado.
UC6	Editar Evento	Neste caso de uso, quando o organizador deseja faze alguma alteração em seu evento após o cadastro.
UC7	Gerar Certificado	Neste caso de uso, o organizador gera o certificado do evento após a conclusão do evento, onde os participante deveram ter cumprido a carga horaria requerida no evento.
UC8	Gerar Comprovante	Neste caso de uso, o participante após faze a inscrição do evento o sistema vai gerar um comprovante de inscrição, confirmando aquele participante no evento onde ele poderá imprimir.
UC9	Inscrever Participante	Neste caso de uso, após faze uma busca pelo evento aparecerá à opção de inscrição para o participante logo depois que ele fizer o login no site Caça Eventos.

5. Visão Lógica

Descreve as classes mais importantes, sua organização em pacotes e subsistemas de serviço, e a organização desses subsistemas em camadas.

5.1. Visão Geral

A visão geral do Caça Eventos é composta por 3 pacotes:

- View: Este não está preocupado em como a informação foi obtida ou onde ela foi obtida, apenas exibe a informação ao usuário.
- Controller: Este determina o fluxo de apresentação servindo como uma camada intermediária, ou seja, o intercessor entre o View e o Model.

- Model: Este é responsável por tudo o que a aplicação irá fazer. Modelando os dados e o comportamento do sistema, bem como preocupa com o armazenamento, manipulação e geração de dados, sendo um encapsulador de dados e de comportamento independente da apresentação (view).

5.2. Pacotes de Design Significativos do Ponto de Vista da Arquitetura

Pacote – View

Classes	Descrição
TelaHome	Interface da página principal do site, mostrando os eventos criados.
TelaLogin	Tela de entrada de dados para acessar o sistema, onde se coloca o e-mail e senha, tanto o participante quanto o organizador.
TelaInscricao	Após escolhe o evento, uma tela de inscrição aparecerá para o participante faze a inscrição.
TelaCadastrarEvento	Tela de cadastro do evento onde o organizador preenche uma ficha de dados para criação do evento.
TelaCriarConta	Interface de criar conta, de usuário seja ele um participante ou organizador, que irão faze parte do sistema.
TelaRecuperarSenha	Tela de recuperação de senha onde o aparece o campo de e-mail confirmação.
TelaDoEvento	Tela que mostra os dados do evento ao faze a busca.

Pacote – Control

Classes	Descrição
ListarEvento	Função que lista todos os eventos cadastrado no sistema que aparecerá para o usuário.
BuscarEvento	Função que faz a busca dos eventos onde estão cadastrados na plataforma caça eventos.
EditarEvento	Função de edição do evento cadastrado caso o organizado queira fazer alguma alteração.
RemoverEvento	Função que remove o evento feito pelo organizado no sistema.
CadastrarEvento	Função de cadastrar evento onde só o organizado faz após ter feito o login.
EditarUsuario	Função de edição de usuário para de algum dado fornecido.
RemoverUsuario	Função de remoção de usuário caso o organizador ou participante queria deletar sua conta do sistema.

CadastrarUsuario	Função que cadastrar os diferentes tipos de usuários seja ele participante ou organizado.

Pacote – Model

Classes	Descrição
Evento	Classe que criação dos eventos no sistema que terá todos os atributos do evento.
Inscrição	Classe de inscrição dos participantes ao se cadastrar no sistema.
Usuário	Classe de do usuário que irão usar o sistema.

6. Visão de Processos

O sistema é gerenciado por meio de processos, esses processos podem ser divididos com base na sua capacidade de influência para o sistema como um todo, podendo ser classificados em dois tipos:

- Processos leves: São processos de baixa importância dentro do sistema, tais como threads de baixa prioridade criadas para processamento paralelo.
- Processos pesados: São processos de alto impacto dentro do sistema como um todo, em que o mau gerenciamento pode comprometer outras áreas do sistema, tais como threads criadas para a interação com o usuário.

7. Visão de Implantação

O sistema é construído na linguagem Python, com o foco voltado para web. A linguagem Python utiliza para executar o sistema. As máquinas interpretam o código e executam as instruções nele contidas diretamente nos sistemas, como no diagrama abaixo:

8. Visão de Implementação

8.1. Visão Geral

Seguindo o padrão arquitetural MVC, o sistema pode ser dividido em 3 camadas : Model, view, controller. Nessas camadas são separadas as classes do código com base em sua influência no sistema. As classes da camada view são responsáveis apenas pelo front-end do sistema. As classes da camada model são as classes atômicas e as classes compostas essenciais para construção do sistema. As classes da camada controller são responsáveis pela construção do back-end do sistema.

8.2. Camadas

8.2.1 View

A camada view é responsável pela interação do usuário com o sistema, nessa camada são realizados etapas como construção de interface de usuário e interação para a configuração das funcionalidades.

8.2.2 Model

A camada model é responsável pelo armazenamento das classes mais atômicas do projeto, nessa camada são realizados ações como a execução básica do projeto e a execução complexa do evento. As classes que compõem essa camada são, Cadastrar Evento e Criar Conta.

8.2.3 Controller

A camada controller é responsável pela execução de algoritmos complexos como interpretação dos dados da interface para o sistema e o controle do ciclo de vida da aplicação.

9. Tamanho e Desempenho

O sistema Caça Eventos é desenvolvido para a plataforma web. A criação de um sistema com muitas classes e arquivos deve ser ponderada de forma a não afetar (ou afetar o mínimo possível) o desempenho do sistema durante a execução.

10. Qualidade

Sabendo que qualidade e seus atributos é a base para as estratégias e decisões de design da arquitetura. O padrão de arquitetura MVC foi à solução mais satisfatória para atender a qualidade esperada do sistema. Sistema este que deverá ser desenvolvido na linguagem Python, visando uma interface que seja interativa e fácil de usar.