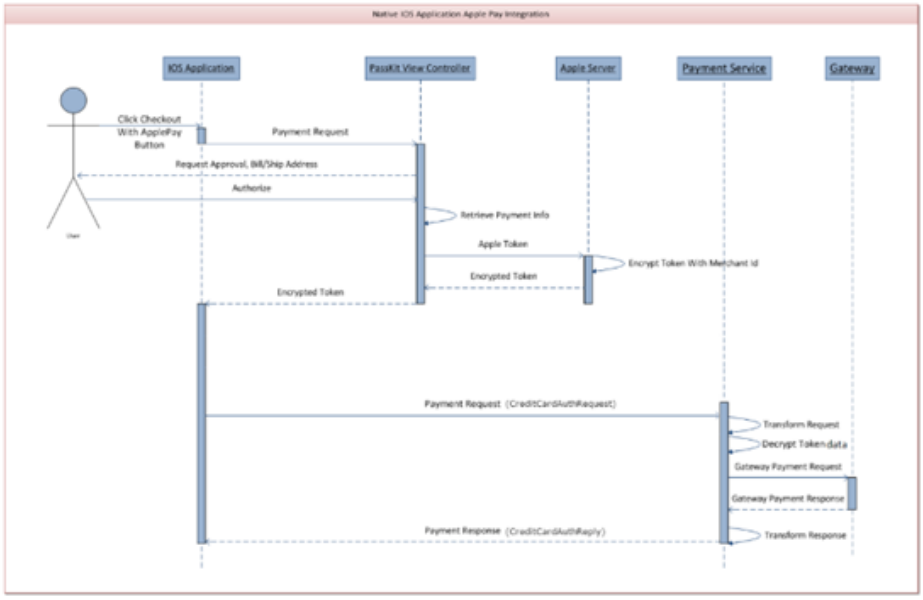
David Pelicano (113391)(pivot), Pedro Melo (114208), Rúben Pequeno (102480), Simão Almeida (113085)

Grupo 501, v2024-03-13.

RELATÓRIO

4 Lab: Modelação de comportamento (interações)

Exercício 4.1



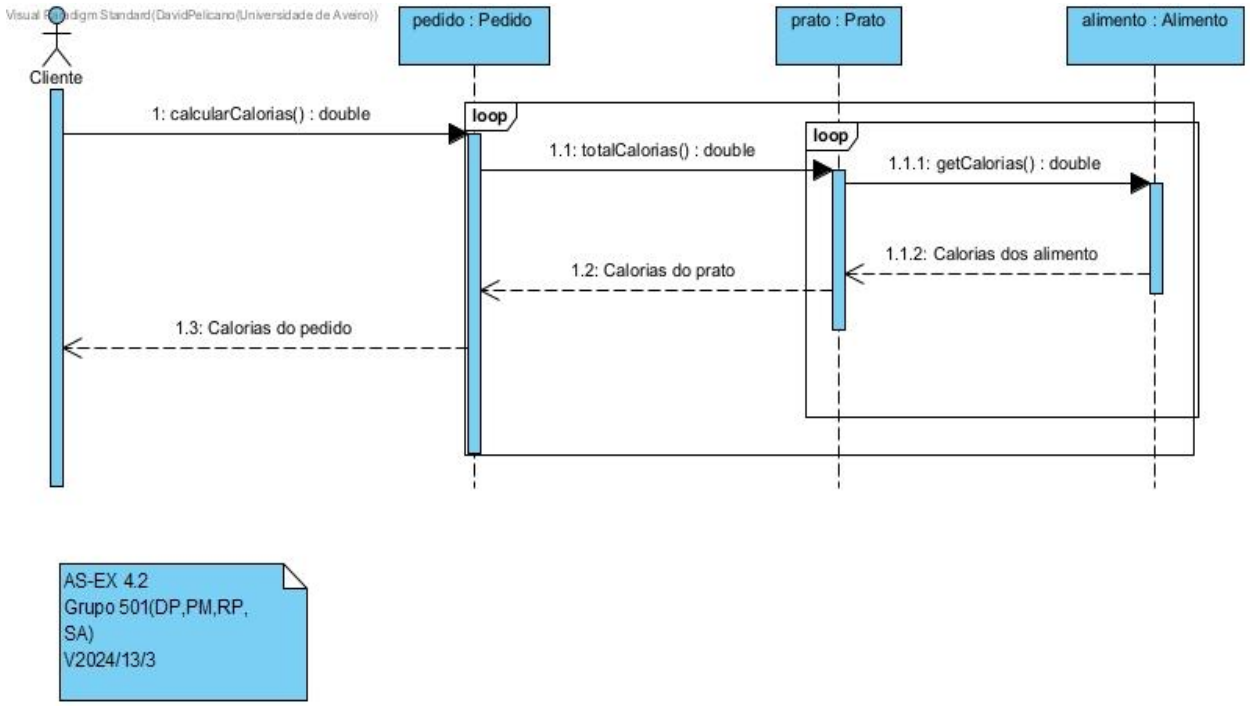
O processo começa quando um utilizador clica no botão “Checkout With ApplePay” numa aplicação iOS. A aplicação, então, envia um pedido de pagamento ao Passkit View Controller, que solicita a aprovação do utilizador e o endereço de envio da fatura ou do produto.

Após a autorização do utilizador, o Passkit View Controller recebe as informações de pagamento do utilizador e é gerado um token. Este é enviado ao servidor da Apple, que encripta o token com um Merchant id e devolve o token encriptado ao Passkit View Controller, que por sua vez devolve á aplicação iOS.

A aplicação iOS envia um pedido de pagamento, que contem o CreditCardAuthRequest, ao serviço de pagamento, que transforma o Request, desencripta o conteúdo do token e faz um pedido de pagamento ao Gateway.

O Gateway processa este pedido e envia de volta uma resposta que é transformada pelo serviço de pagamento e enviada de volta à aplicação iOS em forma de uma resposta de pagamento que contem a CreditCardAuthReply.

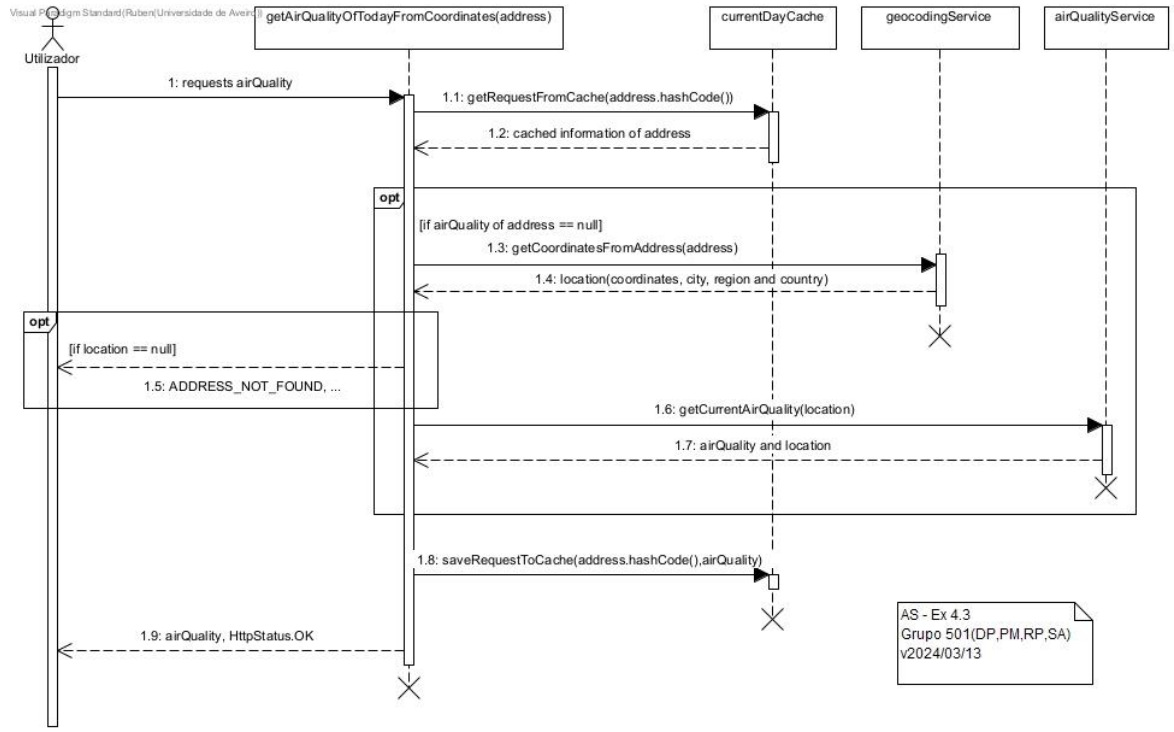
Exercício 4.2



Para este diagrama sequencial temos o ator que neste caso é o Cliente,e 3 lifelines: Pedido,Prato e Alimento, onde o cliente solicita a operação calcularCalorias() ao lifeline Pedido, o Pedido inicia um loop para calcular as calorias totais, solicitando ao lifeline Prato que calcule aas calorias do Prato. O lifeline Prato também inicia um loop e solicita ao lifeline Alimento que retorrne as calorias dos alimentos individuais. As calorias dos alimentos são então retornadas ao lifeline Prato, que calcula as calorias totais do prato e retorna este valor ao lifeline Pedido. FInalmente o pedido calcula as calorias todas do pedido e retorna este valor ao cliente.

Exercício 4.3

a)



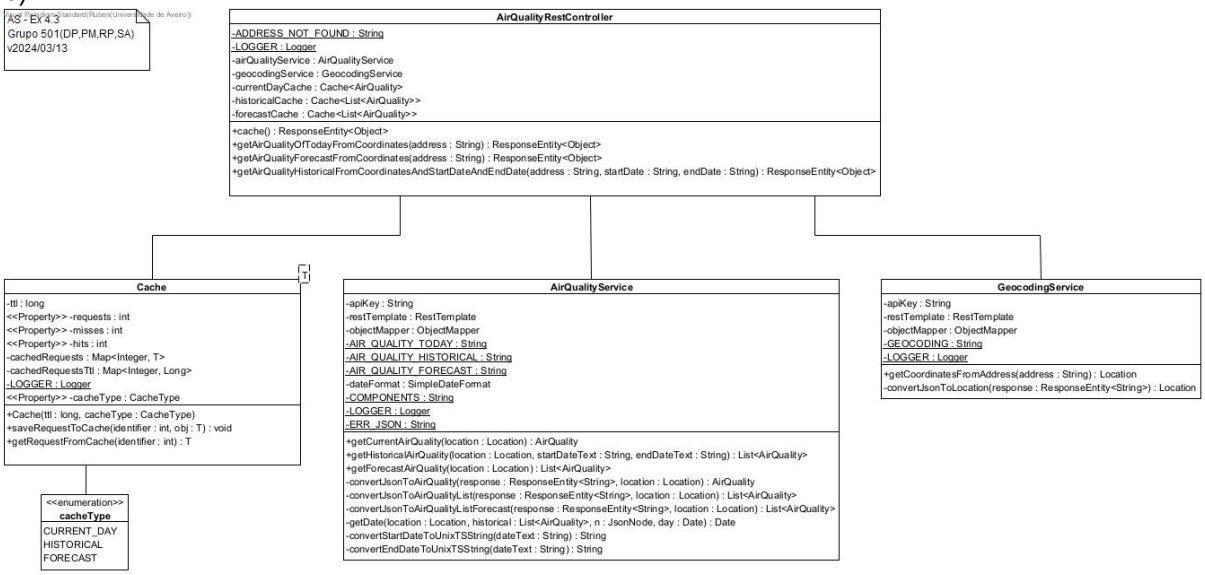
O diagrama de sequência apresentado acima ilustra o processo percorrido a partir do momento que o método *getAirQualityOfTodayFromCoordinates()* é chamado até este devolver uma resposta.
O método começa por invocar getRequestFromCache() ao objeto ***currentDayCache*** para verificar se já existe informação sobre o endereço pedido.

Se já houver informação da qualidade do ar guardada na cache, apenas invoca o método *saveRequestToCache()* para guardar a informação atualizada na cache e acaba por retornar a qualidade do ar, assim como o *HttpStatus.OK(200)*.

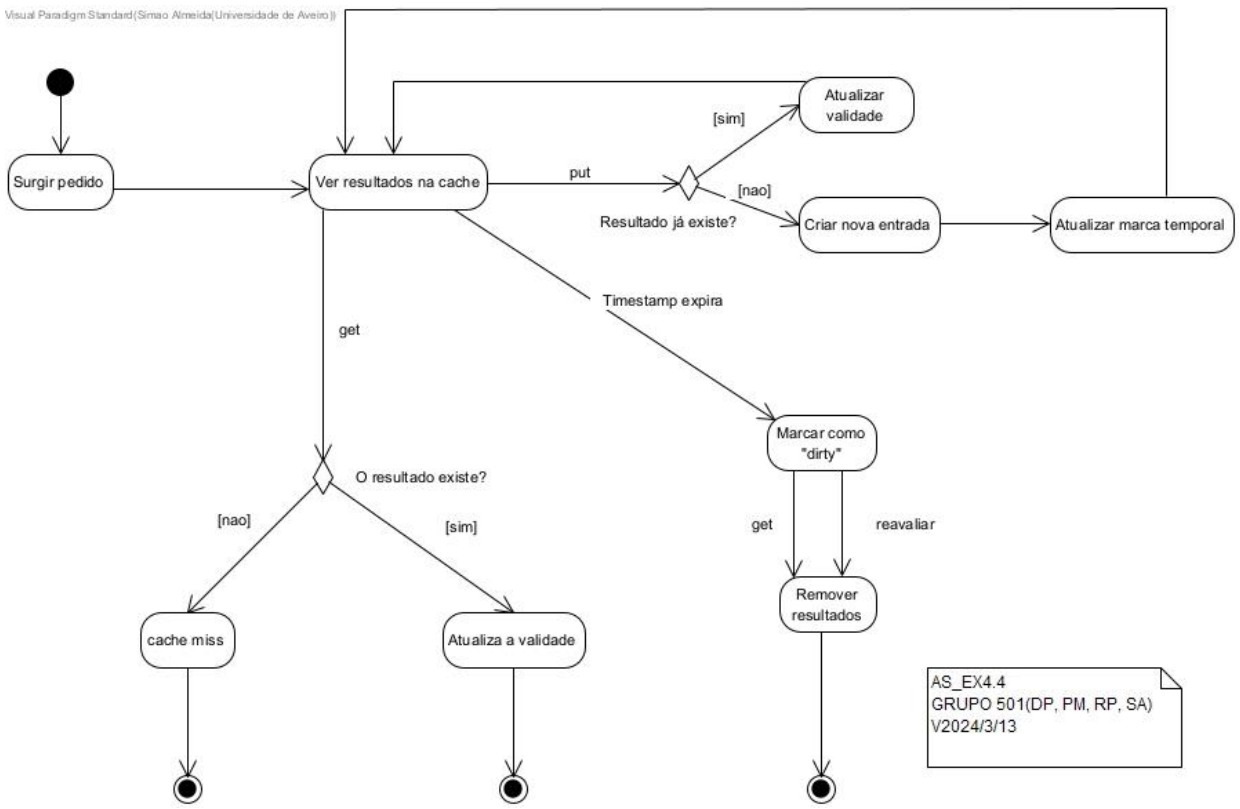
Caso contrário, o método pede as coordenadas do endereço indicado(*getCoordinatesFromAddress()*) ao objeto ***geocodingService***. Se a resposta for nula, o método *getAirQualityOfTodayFromCoordinates() irá terminar a sua execução devolvendo ADDRESS_NOT_FOUND e HttpStatus.NOT_FOUND(404). No caso de a resposta ser válida, é feito um pedido ao objeto* ***airQualityService*** *para obter a qualidade do ar atual(getCurrentAirQuality()).*

Após a obtenção do valor da qualidade do ar, procede ao processo descrito previamente como se a informação estivesse sido obtida pela cache.

b)



Exercício 4.4

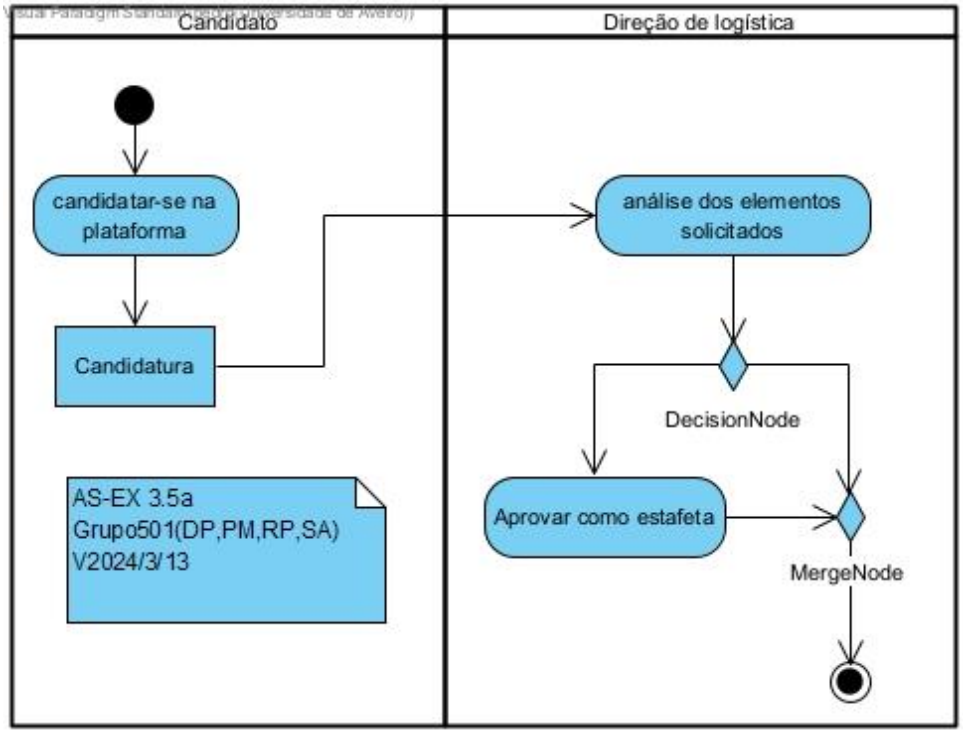


Este diagrama de atividade mostra como o projeto utiliza a cache para otimizar as suas operações. O projeto guarda os resultados da previsão da qualidade do ar na cache durante um certo intervalo de tempo. Surgindo um

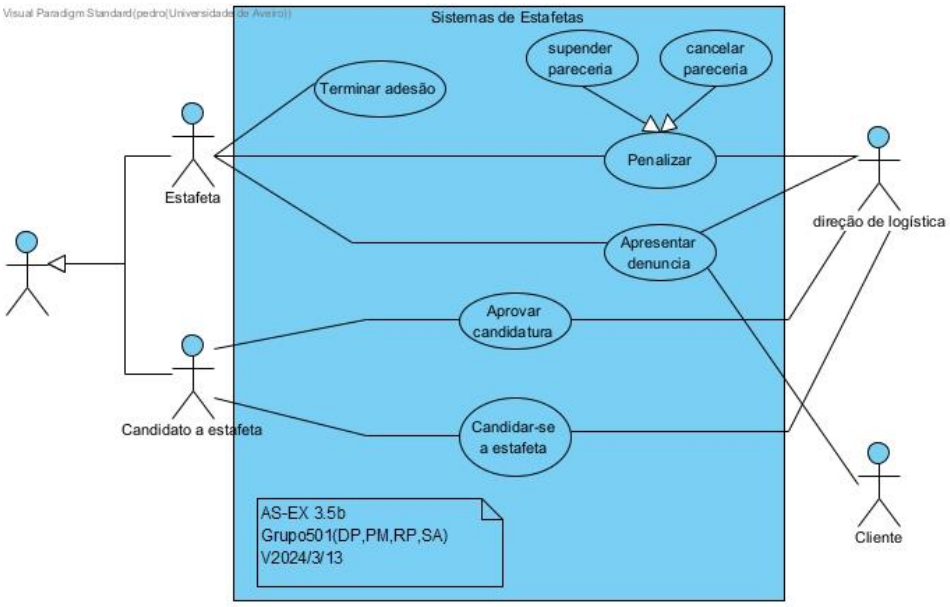
pedido “idêntico”, é respondido com os resultados anteriores guardados na cache. Através de um put, é possível criar uma entrada para um resultado. Se o resultado já existe, então a validade é atualizada. Através de um get é possível atualizar a validade de uma entrada. Se o resultado do get não existir, leva a um “cache miss”. Se o timestamp expirar, pode ser removido através do método get. Periodicamente as entradas também são reavaliadas e as inválidas são removidas.

Exercício 4.5

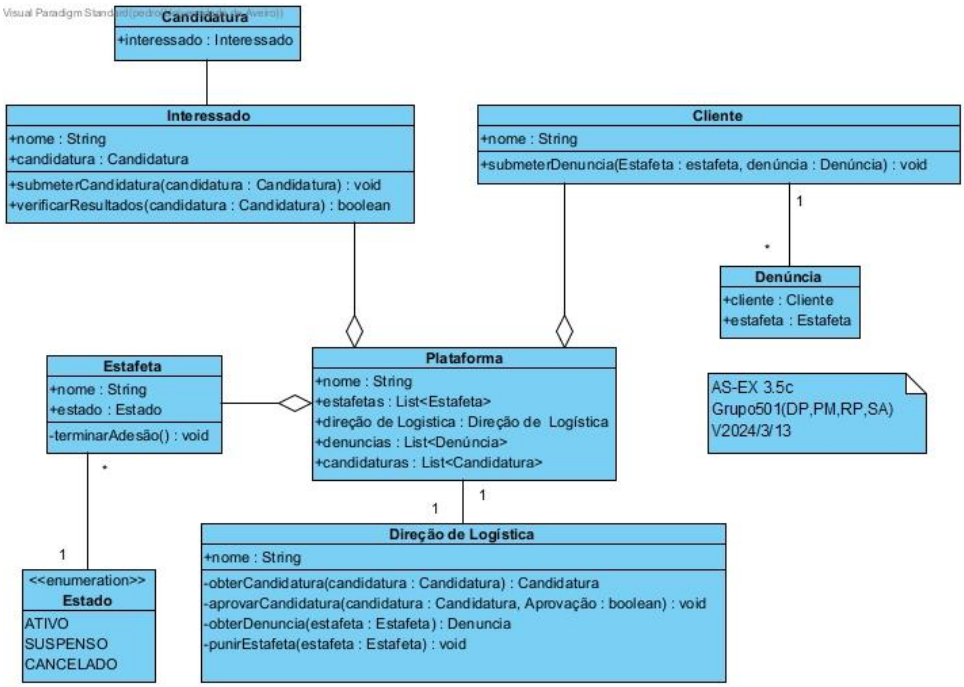
a. **Modelo de atividades**: Este modelo pode ser usado para representar o processo de adesão de um estafeta à plataforma, desde a candidatura espontânea até a aprovação pela direção de logística.



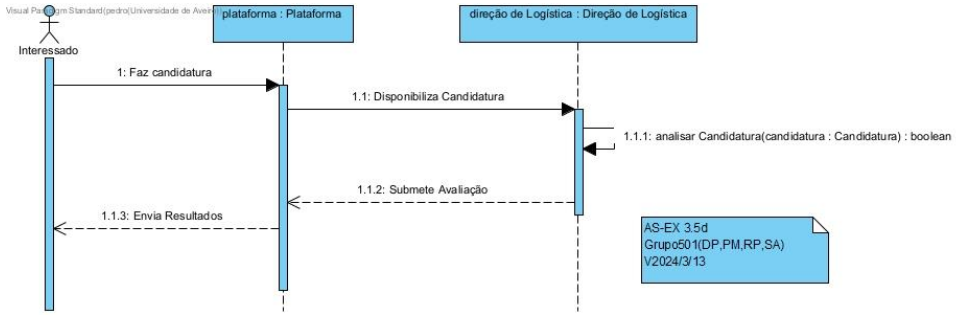
b. **Modelo de casos de utilização**: Este modelo pode ser usado para representar as interações entre os atores (interessados, estafetas, direção de logística, clientes) e o sistema.



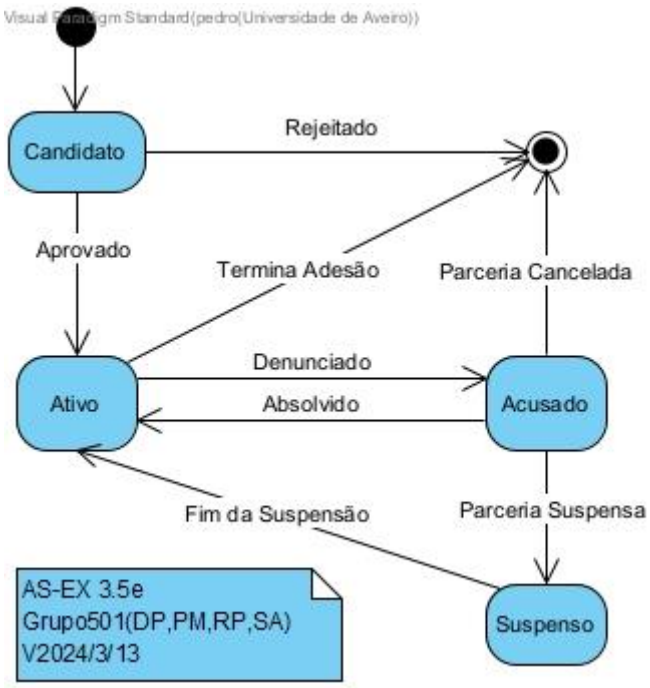
c. **Modelo do domínio (com o diagrama de classes)**: Este modelo pode ser usado para representar as entidades envolvidas (Estafeta, Plataforma, Direção de Logística, Cliente e Interessado) e seus relacionamentos.



d. **Modelo de interação (com o diagrama de sequência):** Este modelo pode ser usado para representar a sequência de interações entre os interessados, plataforma e direção de Logística e o sistema durante o caso de uso da candidatura.



e. **Modelo de (máquina de) estados (com o diagrama de estados):** Este modelo poderia ser usado para representar os diferentes estados de um estafeta na plataforma (candidato, ativo, suspenso, acusado) e as transições entre esses estados.



A relevância de criar estes modelos é que ajudam a entender melhor o sistema e podem servir como documentação para futuras referências. Além disso, eles podem ajudar a identificar possíveis problemas ou melhorias no sistema.