MINISTÉRIO DA EDUCAÇÃO

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Campus Santa Mônica

CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GUILHERME ALVES CARVALHO

YURI HENRIQUE BERNARDES MACIEL

MODELAGEM DE SOFTWARE

DRIVE IN PARK

UBERLÂNDIA-MG

MAIO/2022

GUILHERME ALVES CARVALHO

YURI HENRIQUE BERNARDES MACIEL

MODELAGEM DE SOFTWARE

DRIVE IN PARK

Trabalho de conclusão de disciplina apresentado como requisito parcial para obtenção de nota na disciplina de Modelagem de Software do Curso de Bacharelado em Ciência da Computação da Universidade Federal de Uberlândia - *Campus* Santa Mônica.

Professor/Orientador: Dr. Marcelo de Almeida Maia

UBERLÂNDIA-MG

MAIO/2022

SUMÁRIO

1 INTRODUÇÃO

O presente documento tem como objetivo especificar um software de gerenciamento de estacionamentos, denominado DriveInPark. A aplicação é voltada para o gerenciamento interno do estabelecimento, realizando o regulamento de entrada e saída de veículos do mesmo, assim como o cadastro de clientes, convênios e recebimento de pagamentos. O sistema também permitirá a emissão de relatórios de faturamento.

O software será altamente desacoplado, isto é, permite que funcionalidades sejam ativadas ou não de acordo com a necessidade e condição do estabelecimento. Desse modo, daremos suporte a produtos WPS ou Parklio, que fornecem hardware para automatização de processos em estacionamento. Entre as funcionalidades de automatização disponíveis, teremos: emissão de tickets via totem, validação de tickets ou cartão de acesso via totem, abertura automática de cancelas, utilização de sensores para disponibilidade de vagas.

O estacionamento deve ter cabines para que funcionários do estacionamento (usuários do sistema) validem os tickets mediante pagamento do cliente, o qual realizou a entrada no estabelecimento sem o uso de um cartão (não é mensalista nem funcionário de empresa conveniada), e sim de um ticket de papel.

É possível também cadastrar cotações, dando flexibilidade para criação de diferentes métodos de cobrança que podem ser selecionados ao realizar o fechamento de cadastros - mensalista ou convênio. Vale ressaltar que ao menos uma cotação deve ser registrada para que cadastros sejam permitidos.

A criação de convênios, permite cadastrar informações de uma empresa e o valor acordado (cotação), de modo que todos os funcionários do local conveniado usufruam o estacionamento em horário comercial sem pagar. Nesse caso o pagamento é feito através da funcionalidade de cobrança automática, emitindo nota fiscal e enviando-a por e-mail.

2 DESENVOLVIMENTO

No capítulo a seguir, iremos detalhar o funcionamento do software e seus requisitos para casos de sucesso e falha. Dessa forma, realizaremos a modelagem utilizando a ferramenta StarUML, a qual tem suporte para a plotagem de diagramas UML (Unified Modeling Language).

2.1 Requisitos funcionais e não funcionais

2.1.2 Funcionais

- Vagas (CRUD - Criar, Ler, Atualizar e Remover);
- Cotações (CRUD);
- Clientes mensalistas ou funcionários com convênio (CRUD);
    - Emissão de cartão para esse cliente;
- Validação de tickets pelo usuário do sistema;
- Gastos por período (CRUD);
- Convênio (empresa - CRUD);
- Emissão de nota fiscal;
- Emissão de relatório de faturamento por período;
- Emissão de relatório de ocupação do estacionamento através dos sensores (vagas ocupadas e vazias por tipo - moto e carro);
- Sensores de presença nas vagas conectadas ao software;

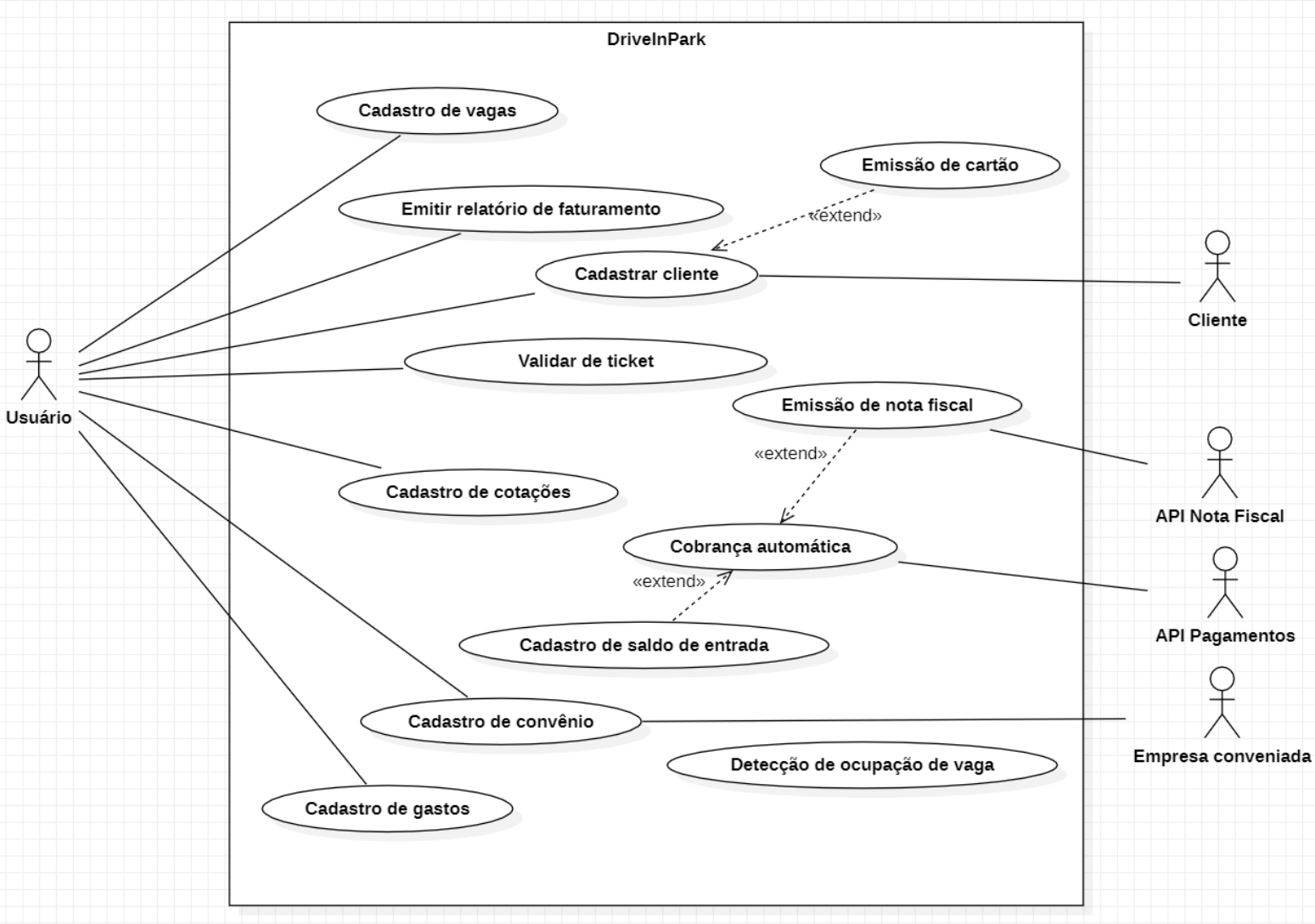
2.1.3 Não funcionais

- Ajuda automatizada para gerenciamento do o estacionamento;
- Integração com hardwares inteligentes para controle do estacionamento;
- Facilidade e segurança para gerenciamento financeiro;
- Facilidade para autenticação e cobrança de uso do serviço;
- Interface agradável e de fácil uso, boa experiência do usuário;
- Software desacoplado de hardwares auxiliares.

2.2 Casos de Uso

Neste capítulo iremos apresentar os diversos casos de uso da aplicação. A seguir temos um diagrama de casos de uso, para uma visão mais abrangente dos mesmos.

Figura 1 - Casos de uso do Drive In Park



Fonte: Autores deste documento

2.2.1 Cadastrar cliente

Nome	Cadastrar cliente.
Descrição	As informações do cliente (mensalista ou funcionário de empresa conveniada) são cadastradas pelo usuário para uso posterior.
Dependência	Nenhum.
Atores	Usuário, Cliente.
Sequência de ações	1. O cliente passa suas informações pessoais para o usuário do sistema. 2. Usuário do sistema digita as informações no software e cria o cliente com seu tipo (mensalista e de convênio). 3. Usuário emite cartão para cliente, para uso posterior.
Caso de sucesso	Cliente é cadastrado e cartão é emitido para cliente.
Caso de falha	Cliente não é criado por inserção incorreta de informações.
Solução do caso de falha	1. Usuário revisa informações inseridas. 2. Encontra erro e preenche novamente. a. Erro pode ser na inserção. b. Erro pode ser na informação passada pelo cliente. 3. O cliente é cadastrado após inserção pelo usuário.
Trigger	Cliente normal deseja se tornar mensalista, ou funcionário de empresa conveniada não cadastrado deseja utilizar o benefício da empresa.
Frequência de uso	Intenso no início para cadastrar a carteira de clientes inicial e médio após o período inicial.

2.2.2 Validar ticket

Nome	Validar ticket
Descrição	Permite que o cliente valide seu ticket em conjunto com o usuário do sistema, de modo que possa sair do estacionamento ao apresentá-lo no terminal de saída.
Dependência	Nenhuma.
Atores	Usuário
Sequência de ações	1. O cliente apresenta seu ticket ao usuário do sistema. 2. O usuário utiliza o scanner conectado ao software e obtém valor a ser cobrado. 3. Usuário realiza cobrança no cliente baseado na cotação da entrada. 4. Usuário insere saída de veículo do estacionamento e libera cliente para retirar seu veículo. 5. Informação de ticket validado é enviado para terminais de saida.
Caso de sucesso	Cliente paga a cobrança e tem seu ticket validado.
Caso de falha	Não há.
Solução do caso de falha	Não há.
Trigger	Após uso do estabelecimento, cliente precisa validar ticket para retirar seu veículo.
Frequência de uso	Diário.

2.2.3 Cadastro de vagas

Nome	Cadastro de vagas.
Descrição	Usuário realiza cadastro de vagas do seu estabelecimento, preenchendo informações sobre a mesma.
Dependência	Nenhuma.
Atores	Usuário.
Sequência de ações	1. Usuário acessa tela de cadastro de vagas. 2. Preenche formulário com informações da vaga: tipo (de carro ou moto), número. 3. Realiza o cadastro com informações preenchidas.
Caso de sucesso	A vaga é cadastrada e pode ser usada no sistema como local de estacionamento.
Caso de falha	O cadastro da vaga não pode ser realizado.
Solução do caso de falha	1. Usuário revisa informações inseridas. 2. Encontra erro e preenche novamente.
Trigger	O usuário, dono do estabelecimento, necessita de cadastrar vagas para utilizar o sistema de sensores.
Frequência de uso	Esporadicamente, geralmente ao configurar o aplicativo pela primeira vez. Também é utilizado caso haja expansão no estabelecimento.

2.2.4 Cadastro de cotações

Nome	Cadastro de cotações.
Descrição	Registros dos valores por hora utilizados para o cálculo do valor cobrado no final da saída do veículo do cliente.
Dependência	Nenhuma.
Atores	Usuário.
Sequência de ações	1. Usuário informa nome da cotação. 2. Usuário informa valor da cotação.
Caso de sucesso	A cotação é cadastrada com sucesso.
Caso de falha	A cotação inserida não é cadastrada.
Solução do caso de falha	1. Usuário revisa nome e contação inserida. 2. Usuário encontra erro e corrige informação inserida de forma errada. 3. O cadastro de cotação é realizado com sucesso.
Trigger	Usuário quer criar uma nova cotação para precificar sua cobrança para clientes comuns, mensalistas ou beneficiários de convênio.
Frequência de uso	Esporadicamente, geralmente ao configurar o aplicativo pela primeira vez. Também é utilizado nos casos de estabelecimento de convênios e alteração de preços.

2.2.5 Cadastro de convênio

Nome	Cadastro de convênio.
Descrição	Usuário cadastra um convênio em parceria com alguma empresa.
Dependência	Cadastro de cotação.
Atores	Usuário, empresa conveniada.
Sequência de ações	1. Usuário deve preencher informações do convênio em conjunto com representante da empresa conveniada: nome empresa, CNPJ, cotação, validade. 2. O cadastro do convênio é realizado no sistema.
Caso de sucesso	O convênio é registrado no sistema e agora usuários podem ser cadastrados como conveniados da empresa ao apresentar comprovante de vínculo empregatício.
Caso de falha	O convênio não é cadastrado.
Solução do caso de falha	1. Usuário revisa nome e cotação inserida. a. Caso não tenha uma cotação para o convênio a ser cadastrado, a cotação deve ser cadastrada. 2. Usuário encontra erro e corrige informação inserida de forma errada. 3. O cadastro de convênio é realizado com sucesso.
Trigger	Uma empresa entra em contato para realizar convênio com o estabelecimento.
Frequência de uso	Esporadicamente.

2.2.6 Emissão de nota fiscal

Nome	Emissão de nota fiscal
Descrição	Sistema emite nota fiscal após realização de pagamento
Dependência	Validar ticket, Cobrança automática
Atores	API Nota Fiscal
Sequência de ações	1. O sistema coleta informações do pagamento realizado. 2. O sistema realiza requisição para a API Nota Fiscal. 3. API retorna PDF com nota fiscal.
Caso de sucesso	API retorna PDF de nota fiscal que pode ser impresso ou enviado por e-mail.
Caso de falha	Erro de requisição com a API.
Solução do caso de falha	1. Caso o erro persista, a emissão de nota é colocada em fila. 2. O sistema, quando possível, realiza as tarefas pendentes na fila.
Trigger	Pagamento feito no sistema.
Frequência de uso	Diariamente, em conjunto com validação de ticket ou cobrança automática

2.2.7 Cobrança automática

Nome	Cobrança automática
Descrição	Emite de forma assíncrona (mensal) cobranças para clientes mensalistas e empresas conveniadas
Dependência	Nenhuma
Atores	API de Pagamentos
Sequência de ações	1. O sistema verifica se há necessidade de gerar cobranças. 2. Caso necessário, o sistema utiliza a API de Pagamentos para gerar cobrança.
Caso de sucesso	Cobrança feita com sucesso e cadastro de saldo de entrada criado
Caso de falha	Erro de requisição com a API.
Solução do caso de falha	1. Caso o erro persista, a emissão de nota é colocada em fila. 2. O sistema, quando possível, realiza as tarefas pendentes na fila.
Trigger	Rotina do sistema.
Frequência de uso	Variável de acordo com a quantidade de convênios e mensalistas cadastrados.

2.2.8 Cadastro de saldo de entrada

Nome	Cadastro de saldos de entrada
Descrição	Sistema cadastra saldo de entrada proveniente de pagamento
Dependência	Validar ticket, Cobrança automática.
Atores	Nenhum
Sequência de ações	1. Uma cobrança feita é detectada, seja automática ou em momento de validação de ticket. 2. Saldo proveniente do pagamento é no banco de dados.
Caso de sucesso	O saldo de entrada é cadastrado no banco de dados.
Caso de falha	Nenhum.
Solução do caso de falha	Nenhum.
Trigger	Uma cobrança é realizada
Frequência de uso	Diariamente, em conjunto com a realização de cobranças

2.2.9 Cadastro de gastos

Nome	Cadastro de gastos
Descrição	O usuário cadastra algum gasto para ser descontado do saldo mensal do estabelecimento.
Dependência	Nenhum
Atores	Usuário
Sequência de ações	1. Usuário acessa seção de faturamento do estacionamento. 2. Usuário insere manualmente um gasto.
Caso de sucesso	Gasto é cadastrado no banco de dados do sistema.
Caso de falha	Nenhum.
Solução do caso de falha	Nenhum.
Trigger	Usuário deseja cadastrar gastos.
Frequência de uso	Esporadicamente, de acordo com desejo ou necessidade de usuário.

2.2.10 Emitir relatório de faturamento

Nome	Emitir relatório de faturamento
Descrição	Usuário emite relatório de faturamento pelo sistema.
Dependência	Cadastro de saldos de entrada, Cadastros de gastos
Atores	Usuário
Sequência de ações	1. Usuário solicita relatório de faturamento. 2. Sistema consulta banco de dados e obtém informações de entrada e saída de capital do mês do corrente. 3. Sistema entrega relatório em arquivo de planilha.
Caso de sucesso	Relatório é gerado para o usuário
Caso de falha	Não há nenhum faturamento
Solução do caso de falha	1. Esperar para que haja dados de faturamento no mês corrente.
Trigger	Usuário deseja emitir relatório de faturamento.
Frequência de uso	Esporadicamente, de acordo com desejo ou necessidade de usuário.

2.2.11 Emissão de cartão

Nome	Emissão de cartão.
Descrição	O sistema utiliza impressora de cartões para imprimir cartão de acesso.
Dependência	Cadastrar cliente
Atores	Nenhum.
Sequência de ações	1. O sistema detecta que cadastro de cliente é feito. 2. O sistema utiliza informações do cadastro e vincula um cartão ao cliente cadastrado. 3. O sistema utiliza impressora de cartões conectada e imprime cartão.
Caso de sucesso	O cliente recebe um cartão para acessar o estacionamento.
Caso de falha	Não há impressora conectada.
Solução do caso de falha	O sistema cancela o cadastro do cliente.
Trigger	O usuário deseja cadastrar um cliente no estacionamento.
Frequência de uso	Variável de acordo com a necessidade dos clientes.

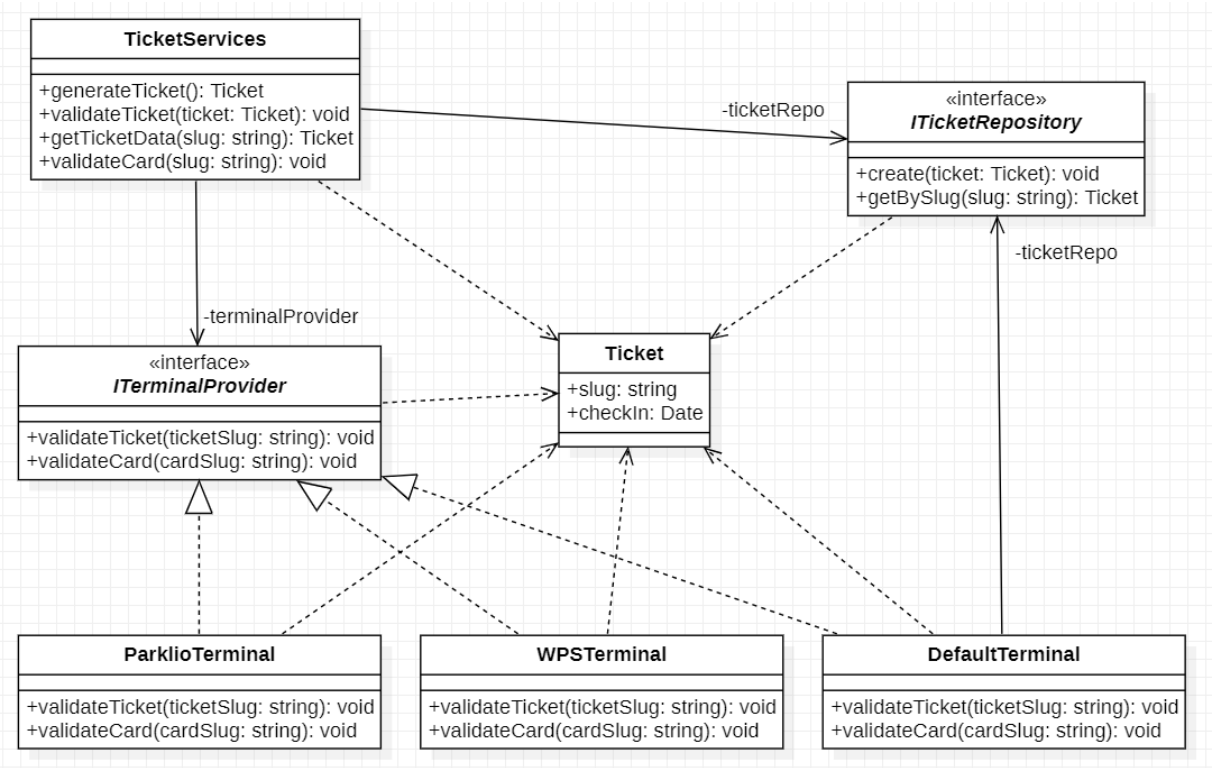
2.2.12 Detecção de ocupação de vaga

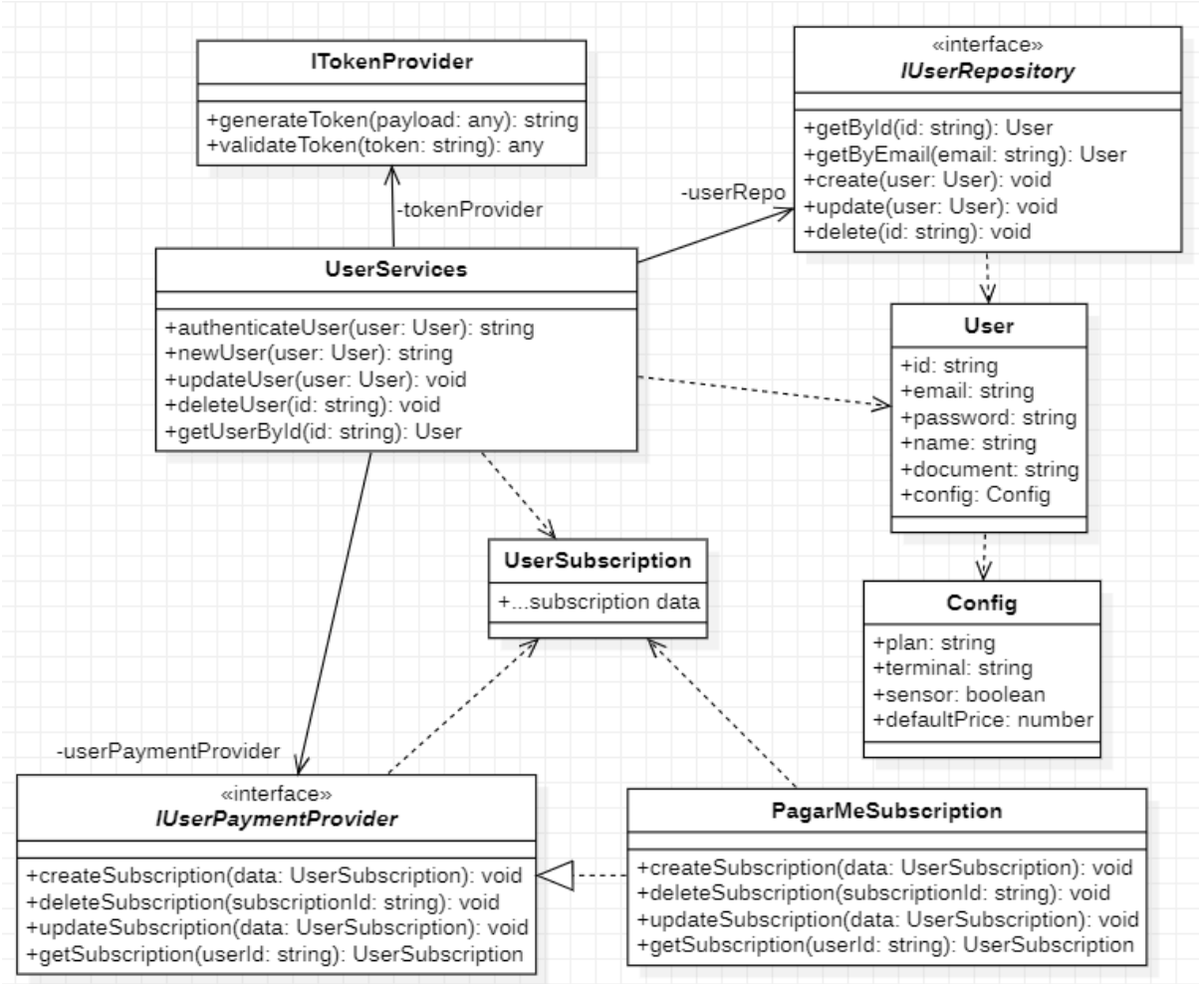
Nome	Detecção de ocupação de vaga.
Descrição	O sistema utiliza os sensores e informações de vagas cadastradas para detectar ocupação do sistema.
Dependência	Cadastro de vagas.
Atores	Nenhum.
Sequência de ações	1. O sistema recebe uma informação do sensor de vaga. 2. O sistema analisa se a vaga está ocupada ou não. 3. O sistema gera relatório de ocupação do sistema.
Caso de sucesso	Informação de ocupação pode ser vista por usuários do sistema ou enviada para um painel externo.
Caso de falha	Não há vagas cadastradas ou sensores configurados.
Solução do caso de falha	A funcionalidade do sistema simplesmente não funcionará.
Trigger	O sensor de vaga envia informação para o sistema.
Frequência de uso	Diariamente.

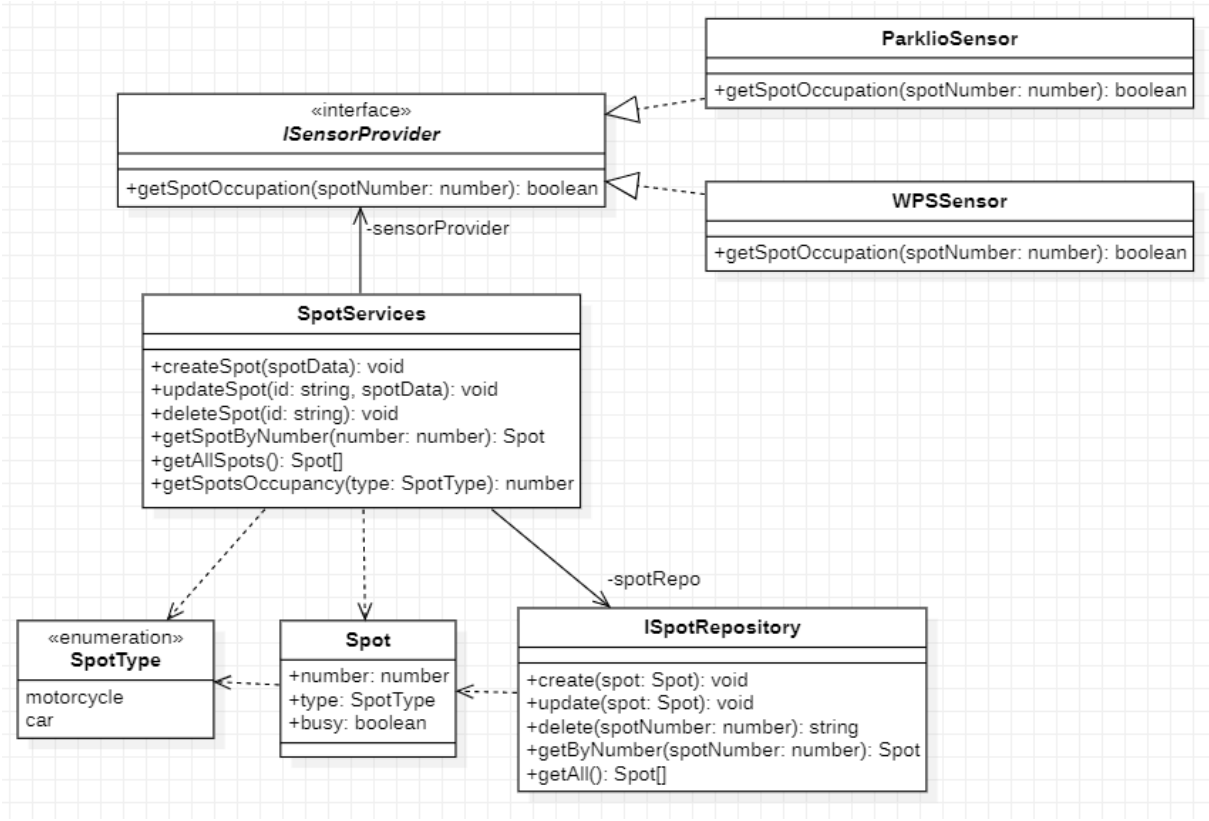
2.3 Diagrama de classe conceitual

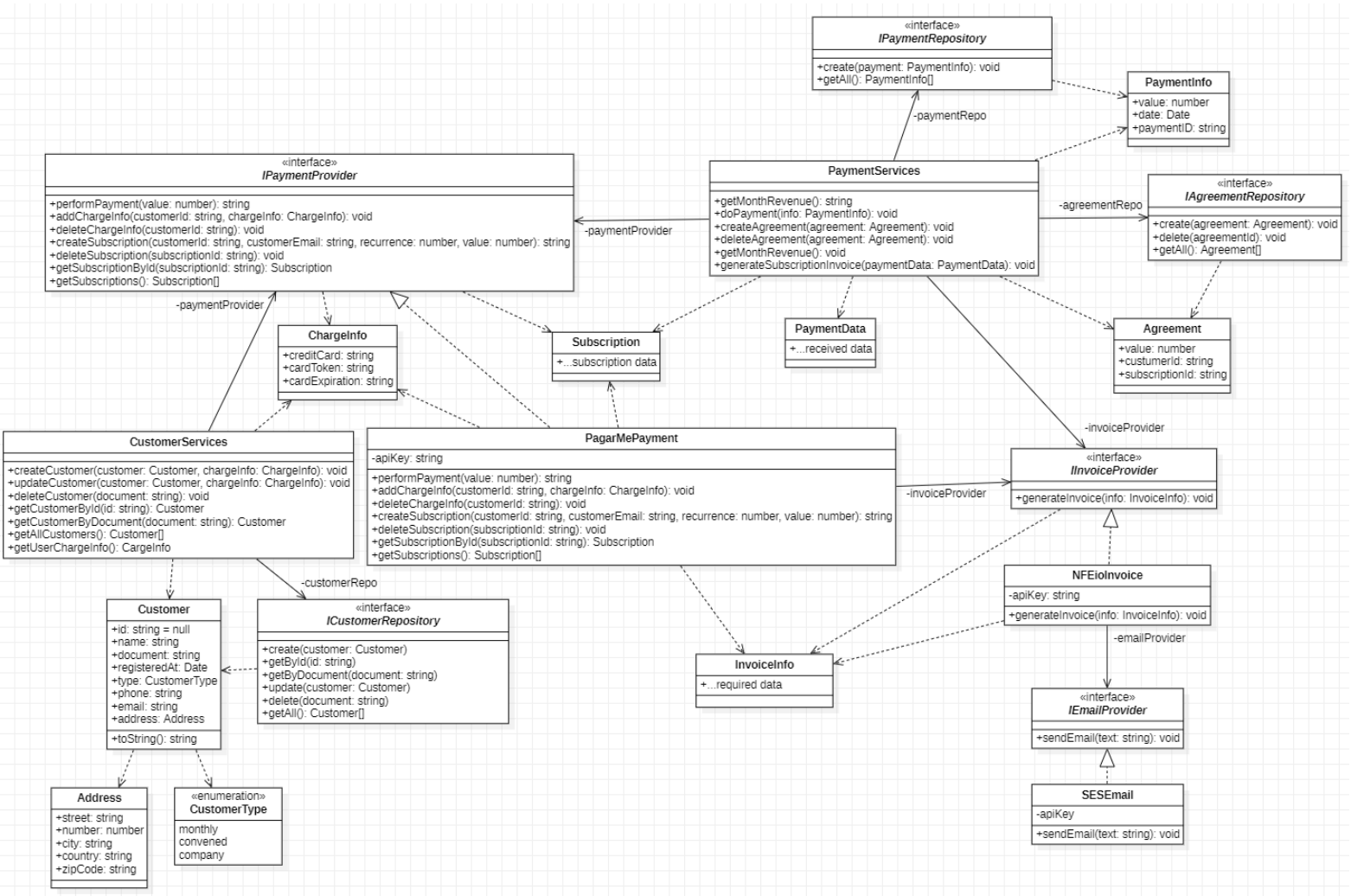
Os diagramas de classe a seguir demonstram a estrutura de classes do backend da aplicação. Para melhor entendimento dos diagramas, tenha em mente que:

- Services são as classes responsáveis por tratar regras de negócios de determinada parte da aplicação.
- Classes Repository são responsáveis pela comunicação com o banco de dados. A interface cria um padrão a ser seguido pelo repositório independente do banco e método de comunicação utilizado.
- Providers são classes responsáveis por realizar chamadas e funções com serviços, bibliotecas ou API de terceiros.




ITokenProvider
+generateToken(payload: any): string
+validateToken(token: string): any
«interface»
IUserRepository
+getById(id: string): User
+getByEmail(email: string): User
+create(user: User): void
+update(user: User): void
+delete(id: string): void
-tokenProvider
-userRepo
UserServices
+authenticateUser(user: User): string
+newUser(user: User): string
+updateUser(user: User): void
+deleteUser(id: string): void
+getUserById(id: string): User
User
+id: string
+email: string
+password: string
+name: string
+document: string
+config: Config
UserSubscription
+...subscription data
Config
+plan: string
+terminal: string
+sensor: boolean
+defaultPrice: number
-userPaymentProvider
«interface»
IUserPaymentProvider
+createSubscription(data: UserSubscription): void
+deleteSubscription(subscriptionId: string): void
+updateSubscription(data: UserSubscription): void
+getSubscription(userId: string): UserSubscription
PagarMeSubscription
+createSubscription(data: UserSubscription): void
+deleteSubscription(subscriptionId: string): void
+updateSubscription(data: UserSubscription): void
+getSubscription(userId: string): UserSubscription

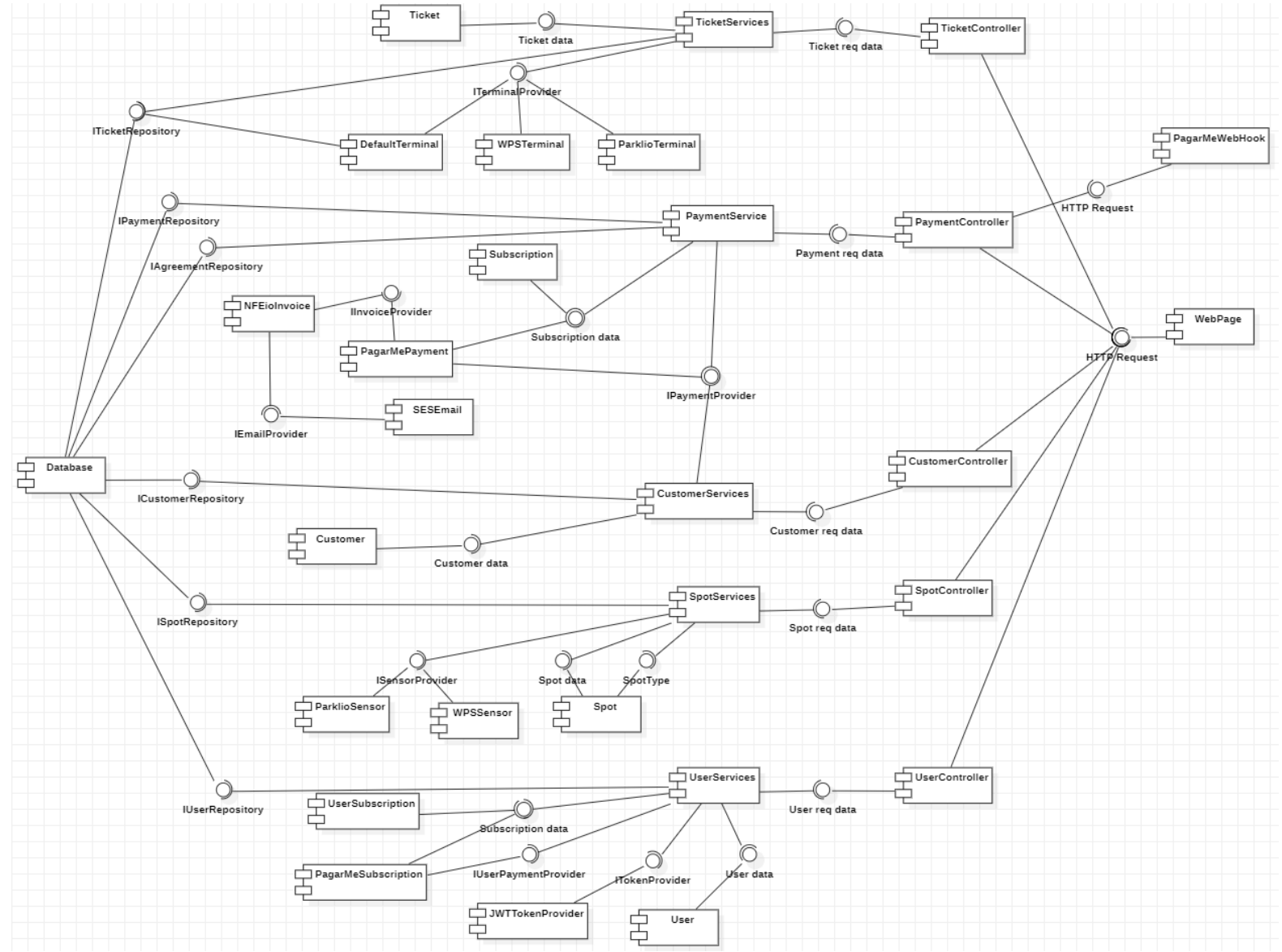

ParklioSensor
+getSpotOccupation(spotNumber: number): boolean
«interface»
ISensorProvider
+getSpotOccupation(spotNumber: number): boolean
WPSSensor
+getSpotOccupation(spotNumber: number): boolean
-sensorProvider
SpotServices
+createSpot(spotData): void
+updateSpot(id: string, spotData): void
+deleteSpot(id: string): void
+getSpotByNumber(number: number): Spot
+getAllSpots(): Spot[]
+getSpotsOccupancy(type: SpotType): number
-spotRepo
«enumeration»
SpotType
motorcycle
car
Spot
+number: number
+type: SpotType
+busy: boolean
ISpotRepository
+create(spot: Spot): void
+update(spot: Spot): void
+delete(spotNumber: number): string
+getByNumber(spotNumber: number): Spot
+getAll(): Spot[]



2.4 Diagrama de componentes

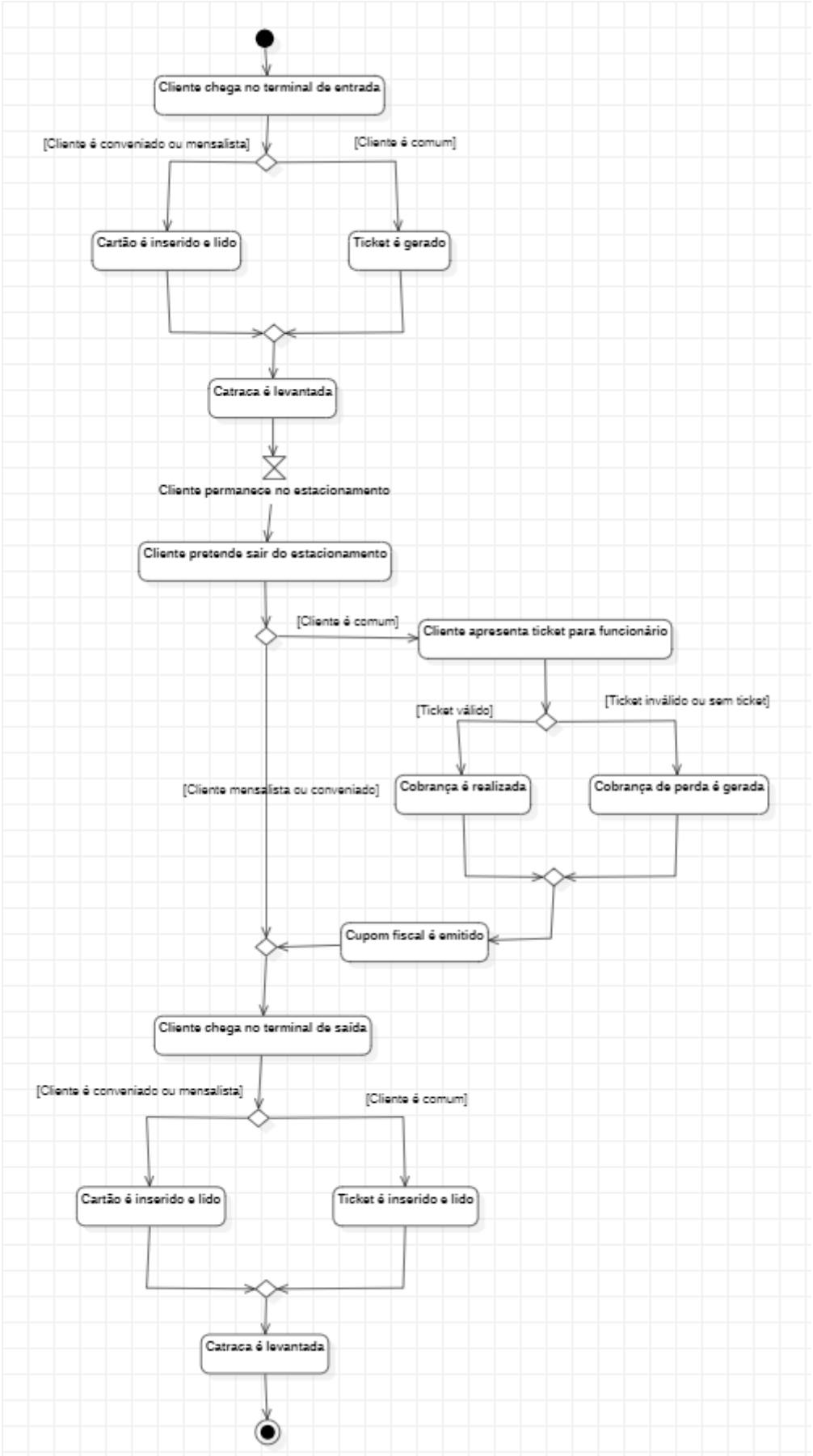
Este diagrama mostra uma visão geral de como as diferentes partes do aplicativo se comunicam entre si. Para melhor entendimento, tenha em mente que:

- Controllers são responsáveis por receber uma requisição do client e tratá-la, garantindo que ela possa ser entendida e processada pelos services.
- WebPage se refere à aplicação front-end do DriveInPark.
- WebHook se refere aos eventos disparados por API's de terceiros para a API do DriveInPark.



2.5 Diagrama de atividade

A seguir temos um diagrama de atividade, mostrando o fluxo pelo qual o cliente do estacionamento deve passar ao utilizar o serviço do estabelecimento.


Cliente chega no terminal de entrada
[Cliente é conveniado ou mensalista]
[Cliente é comum]
Cartão é inserido e lido
Ticket é gerado
Catraca é levantada
Cliente permanece no estacionamento
Cliente pretende sair do estacionamento
[Cliente é comum]
Cliente apresenta ticket para funcionário
[Ticket válido]
[Ticket inválido ou sem ticket]
[Cliente mensalista ou conveniado]
Cobrança é realizada
Cobrança de perda é gerada
Cupom fiscal é emitido
Cliente chega no terminal de saída
[Cliente é conveniado ou mensalista]
[Cliente é comum]
Cartão é inserido e lido
Ticket é inserido e lido
Catraca é levantada

2.6 Estudo de viabilidade

2.6.1 Operacional

Em relação aos aspectos físicos e necessários para que o software ocorra, entra em questão os seguintes pré-requisitos que a organização final necessita, em termos de infraestrutura do estacionamento que utilizará o software:

- Estabelecimento locado ou comprado para implantação do sistema;
- Cabine com computador para instalação do software;
- Funcionário para operar o software;
- Rede Ethernet (software utiliza APIs);
- Entrada e saída com terminal ligado a cancela;
- Vagas com sensores;
- Impressora de cartões e notas fiscais;
- Máquina de pagamento.

Para que o software opere com todas as suas funcionalidades, os itens descritos acima necessitam de estar em funcionamento e conectados ao software.

2.6.2 Econômica

Para o funcionamento do software deveremos custear as APIs consumidas para emissão de nota fiscal. Entretanto, não conseguimos obter a precificação desse serviço, pois é necessário agendar um atendimento para isso.

Analisando software de terceiros no mercado, a maioria cobra planos mensais que entregam para o cliente apenas um software responsável por realizar o gerenciamento do estacionamento, cobrando valores que variam entre R$90-100.

Como nosso software disponibiliza integrações opcionais com dispositivos auxiliares para o estacionamento (através de uma empresa terceirizada), iremos oferecer dois tipos de planos para os nossos clientes:

- Plano básico: R$95/mês
    - Você obtém um software de gerenciamento completo, com emissão de tickets, validação, pagamentos automatizados, cadastro de clientes e emissão de relatórios.
- Plano PRO: R$200/mês
    - Você obtém todas as funcionalidades do plano básico mais a capacidade de utilizar as funcionalidades de automatização, compatível com equipamentos WPS ou Parklio.

2.6.3 Técnica

Para a implantação do software com uso de terminais de entrada e saída, e sensores de presença de ocupação das vagas, é necessário que o estabelecimento utilize produtos WPS ou Parklio. Ambos nos permitem implementar o software de maneira desacoplada pois fornecem APIs para utilização de seus produtos de hardware.

Quanto ao sistema de pagamento, para a cobrança automática utilizaremos a API do Pagar.me. Para o pagamento físico, permitiremos a utilização do software em conjunto com as máquinas de cartão mais utilizadas do mercado, como Mercado Livre, PagSeguro e Cielo, pois estas possuem APIs para integração.

Para a emissão automática de notas fiscais, utilizaremos o serviço NFE.io que fornece uma API completa para consulta e emissão das mesmas.