Relatório de Projeto

Inteligência Artificial para Sistemas Autónomos

Licenciatura em Engenharia Informática e de Computadores

Autores:	**Francisco Engenheiro**
ISEL ID:	49428
Professor:	**Phd Luís Morgado**
Ano académico:	2023/2024

Índice

Capítulo 1

Introdução

Este é o início do capítulo. Exemplo de indentação do segundo parágrafo.

1.1 Nome da secção deste capítulo

Texto da secção. Na figura 1.1 mostra-se o logótipo do ISEL. Em [**?**] encontra várias referências para o assunto. O artigo [**?**] é o mais popular conforme indicação do IEEE. Logo a seguir aparece [**?**]. A identificação das referências deve ser melhorada.

Figura 1.1: Legenda da figura com o logótipo do ISEL.

Continuação do texto depois do parágrafo que refere a figura.

1.2 A segunda secção deste capítulo

Na segunda secção deste capítulo, vamos abordar o enquadramento, o contexto e as funcionalidades.

1.2.1 A primeira sub-secção desta secção

As sub-secções são úteis para mostrar determinados conteúdos de forma organizada. Contudo, o seu uso excessivo também não contribui para a facilidade de leitura do documento.

1.2.2 A segunda sub-secção desta secção

Esta é a segunda sub-secção desta secção, a qual termina aqui.

1.3 Organização do documento

O restante relatório encontra-se organizado da seguinte forma.

Capítulo 2

Enquadramento Téorico

A inteligência artificial, um ramo da engenharia informática, concentra-se no desenvolvimento de algoritmos e sistemas que imitam a capacidade humana de raciocínio. Entre as suas diversas aplicações, destacam-se a visão computacional, o processamento de linguagem natural, a robótica e a aprendizagem automática (machine learning). No contexto da aprendizagem automática, mais concretamente para o desenvolvimento de sistemas autónomos, ao qual este projeto se insere, destacam-se os conceitos de inteligência e autonomia. A inteligência caracteriza-se pela relação entre a cognição (i.e. capacidade de realizar a ação adequada dadas as condições do ambiente) e a racionalidade (i.e., capacidade de decidir no sentido de conseguir o melhor resultado possível perante os objectivos que se pretende atingir e as condições do ambiente). Já a autonomia é a habilidade de um sistema operar de forma independente, sem intervenção externa, seja ela humana ou de outro sistema. Representa uma característica da inteligência e portanto ser autónomo não significa ser inteligente, no entanto, ser inteligente implica ser autónomo.

2.1 Agente Inteligente

O modelo de um sistema inteligente é uma abstracção que permite representar a interacção de um sistema com o seu ambiente. Este implementa um ciclo realimentado percepção-processamento-acção (ver figura 2.1), através do qual é realizado o controlo da função do sistema de modo a concretizar a finalidade desse sistema (e.g., a travagem automática de um automóvel quando deteta um obstáculo à sua frente).

Além da autonomia, as caracteristicas principais de um agente inteligente, que estão dependentes do tipo de arquitetura (ver secção 2.4), são:

- **Reactividade**: Capacidade de reagir aos diversos estímulos do ambiente;
- **Pro-actividade**: Capacidade de tomar a iniciativa em função dos seus objetivos;
- **Sociabilidade**: Capacidade de interagir com outros agentes em prol de atingir os seus objetivos, individuais ou coletivos;

- **Finalidade**: Propósito que o agente deve atingir e ao qual todas as suas características contribuem para a sua concretização.

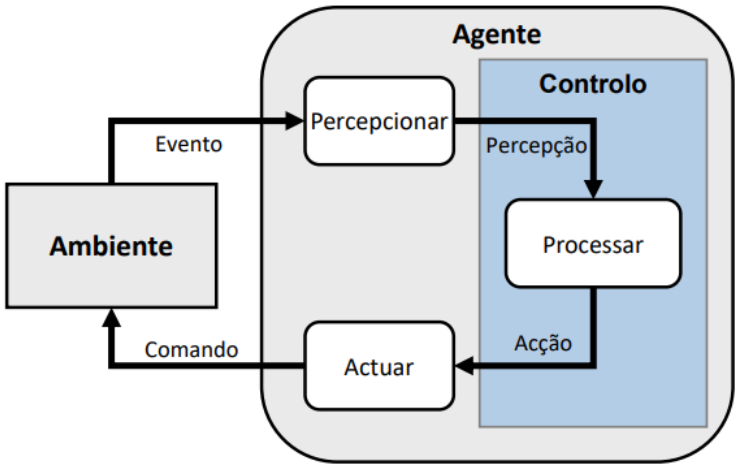

Figura 2.1: Representação conceptual da relação entre agente e ambiente.

2.2 Ambiente

O espaço onde um agente opera é designado por ambiente. É caracterizado (ver figura 2.2) pelas seguintes propriedades:

- **Físico vs. Virtual**: Um ambiente pode existir fisicamente (e.g., uma sala de aula) ou ser simulado de forma virtual (e.g., um jogo de computador);
- **Totalmente Observável vs. Parcialmente Observável**: Um ambiente é totalmente observável se o agente tem acesso a uma descrição completa do ambiente, que lhe permite resolver o problema em questão sem necessitar de guardar estado interno; caso contrário, é parcialmente observável;
- **Tipo de Agente**: Os ambientes podem ser de agente único (i.e., apenas existe um agente a atuar) ou multi-agente (i.e., existem vários agentes a atuar, iguais ou diferentes entre si);
- **Determinístico vs. Estocástico**: Um ambiente é determinístico se o estado seguinte é unicamente determinado pelo estado actual e pela acção do agente; caso contrário, é estocástico. Mais ainda, dizemos que um ambiente continua a ser determinístico, mas mais concretamente estratégico, caso estejamos num ambiente multi-agente onde o próximo estado pode estar também dependente das ações de outros agentes (e.g., num jogo de xadrez);

- **Episódico vs. Sequencial**: Um ambiente é episódico se a experiência do agente é dividida em episódios independentes; caso contrário, é sequencial (i.e., pode ser representado por uma sequência de estados);
- **Estático vs. Dinâmico**: Um ambiente é estático se não se altera enquanto o agente está a tomar uma decisão; caso contrário, é dinâmico;
- **Discreto vs. Contínuo**: Um ambiente é discreto se o número de estados possíveis é finito; caso contrário, é contínuo.

Ambiente	Observável?	Agentes?	Determinístico?	Episódico?	Estático?	Contínuo?
Palavras Cruzadas	Completamente Observável	Agente Único	Determinístico	Sequencial	Estático	Discreto
Poker	Parcialmente Observável	Multi-Agente	Estocástico	Sequencial	Estático	Discreto
Mundo Real	Parcialmente Observável	Multi-Agente	Estocástico	Sequencial	Dinâmico	Contínuo

Figura 2.2: Exemplos de caracterização de ambientes. Retirado de [**?**].

2.3 Agente Relacional

Um agente racional é um tipo de agente que realiza as ações corretas, ou seja, deve conseguir, a partir da exploração e aprendizagem, descobrir estratégias para chegar ao seu objetivo de forma ótima. Para esse efeito, é escolhida a ação que maximiza o valor esperado da medida de desempenho segundo a informação que lhe é fornecida (i.e., percepções) e o conhecimento adquirido até ao momento (e.g., conhecimento disponível sobre o ambiente).

O conceito de recolha de informação entra neste contexto, visto que a exploração pode ser feita fora do âmbito do objetivo com vista a adquirir conhecimento (e.g., estado do ambiente, ações possíveis, recompensas associadas a cada ação, etc) e, assim, melhorar a tomada de decisão seguinte.

No entanto, a racionalidade não implica clarividência, isto é, a ação tomada pode não resultar no que pretendemos ou esperamos, visto que o raciocínio nem sempre leva ao sucesso (e.g., o planeamento de uma viagem de avião, que foi reservada com antecedência e com atenção às condições climatéricas, não garante que o voo não seja cancelado no dia da viagem ou a viajem não seja interrompida por qualquer motivo alheio, mesmo tendo sido tomadas todas as precauções possíveis com base na informação disponível).

2.4 Arquiteturas de Agente

A arquitetura de um agente é a estrutura que define quais os componentes do agente e a forma como estes interagem entre si. Um dos componentes de um agente é o módulo de controlo,

que é responsável por processar percepções e gerar ações. A forma como este módulo é implementado determina o modelo de arquitetura do agente, que pode ser:

- **Reativo**: Associado a um paradigma comportamental, é caracterizado por associações diretas entre perceções e ações. A finalidade deste modelo é a concretização de objetivos implícitos, presentes nas associacões estímulo-resposta;
- **Deliberativo**: Associado a um paradigma simbólico, é caracterizado pela existência de um módulo de deliberação que se situa entre a perceção e a ação. É neste módulo que ocorre o raciocínio e a tomada de decisão, com base em objetivos explícitos;
- **Híbrido**: Combinação dos modelos reativo e deliberativo, com o objetivo de tirar partido das vantagens que cada um oferece.

Capítulo 3

Projeto - Parte 1

Esta parte do projeto incidiu principalmente sobre desenvolvimento de uma biblioteca, em Java, para providenciar abstrações dos subsistemas que representam conceito gerais de Inteligência Artificial (e.g., agente, ambiente) e outros conceitos relacionados (e.g., máquina de estados).

Para tal, foi necessário definir uma arquitetura de software que permitisse a implementação dos diferentes subsistemas de forma independente e modular, seguindo as diretrizes (i.e., métricas (ver secção 3.1.1) e princípios (ver secção 3.1.2)) que garantem a qualidade da arquitetura.

Associado à elaboração da arquitetura de software, foi necessário definir um processo de desenvolvimento de software (ver secção 3.2) que permitisse a implementação dos diferentes subsistemas de forma progressiva, através de diferentes níveis de abstracção (i.e., modelo, arquitetura e implementação).

A implementação da biblioteca foi feita com base na consulta e compressão de diagramas UML e de sequência de forma a garantir a correta implementação dos diferentes subsistemas.

3.1 Arquitetura de software

A arquitetura de software aborda a complexidade inerente ao desenvolvimento de software por meio de uma série de vertentes que estão interligadas.

3.1.1 Métricas

As métricas são medidas de quantificação da arquitectura de um software indicadoras da qualidade dessa arquitectura;

O acomplamento é uma métrica inter-modular que mede o grau de interdependência entre os módulos de um sistema. Pode ser medido através da:

- **Direção**: Unidirecional vs Bidirecional (uni representa menos acoplamento);
- **Visibilidade**: Quando menor for a visibilidade de um módulo, menor é o seu acoplamento;

- **Ordem**: (de menos acoplamento para mais) Herança → Composição → Agregação → Associação → Dependência.

A coesão é uma métrica intra-modular que determina o nível de coerência funcional de um subsistema/módulo, seja pela sua organização (i.e., cada modulo está organizado por conteúdo) ou pela sua funcionalidade (e.g., single responsibility principle - cada modulo tem uma única responsabilidade).

3.1.2 Princípios

Os princípios no contexto da arquitectura de software são um conjunto de convenções que orientam a sua definição, garantindo a qualidade de produção da mesma. Alguns exemplos são:

- **Abstração**: Define a forma como os componentes de um sistema são representados, permitindo a ocultação de detalhes de implementação;
- **Modularização**: Ao qual está associado a decomposição (e.g, divisão do sistema em sub-módulos) e o encapsulamento (i.e., ocultação de detalhes de implementação e/ou manutenção de estado privado e interno);
- **Factorização**: Onde a arquitectura é dividida em camadas, cada uma com um conjunto de responsabilidades bem definidas. Pode ser estrutural (e.g, Herança) e Funcional (e.g, Delegação);

3.2 Processo de Desenvolvimento de Software

O processo de desenvolvimento de software consiste na criação da organização de um sistema de forma progressiva, através de diferentes níveis de abstracção:

- **Modelo (Conceptual)**: Representação abstrata do sistema, que define o que o sistema deve fazer, sem especificar como;
- **Arquitetura (Modelo Concreto)**: Representação concreta do sistema, que define como o sistema deve ser implementado;
- **Implementação**: Código fonte que implementa o sistema definindo como o sistema deve ser executado.

Consiste num processo iterativo, em que as diferentes actividades de desenvolvimento são alternadas ao longo do tempo em função do conhecimento e do nível de detalhe envolvido. Essa alternância poderá ser circular (i.e., implementação → arquitectura → modelo → implementação).

3.2.1 Tipos de Implementação

Tabela 3.1: Tipos de Implementação

Tipo	**Modelo Associado**	**Designação**
Estrutural	UML	Define a estrutura de um sistema, ou seja, a forma como os componentes se relacionam entre si.
Comportamental	Diagrama de Sequência	Define o comportamento de um sistema, ou seja, a forma como os componentes interagem e comunicam entre si.

Mais detalhadamente, os diagramas de sequência ou atividade representam o fluxo de controlo de um sistema, ou seja, a sequência de atividades que um sistema executa e a sua ordem. Definem-se como modelos de interação com uma organização bidirecional (i.e., horizontal $\rightarrow$ tempo e vertical $\rightarrow$ estrutura) e são compostos por diferentes elementos de modelação (e.g., mensagens, operadores, linha de vida).

Já a linguagem de modelação unificada (UML) representa um modelo de comportamento com interação como perspetiva principal de modelação. Este tipo de modelação descreve a forma como as partes de um sistema interagem entre si e com o exterior para produzir o comportamento do sistema. No contexto do projeto, esta linguagem também ajudou a compreender os conceitos fundamentais associados a um sistema computacional (ver secção 3.3) através da representação de diagramas de transição de estado.

3.3 Modelação de um Sistema Computacional

Um sistema computacional geral, pode ser caracterizado de forma abstrata (ver figura 3.1), através de uma representação de estado, que define em cada momento a configuração interna do sistema, e de uma função de transformação que gera as saídas e o próximo estado em função das entradas e do estado actual do sistema [**?**].

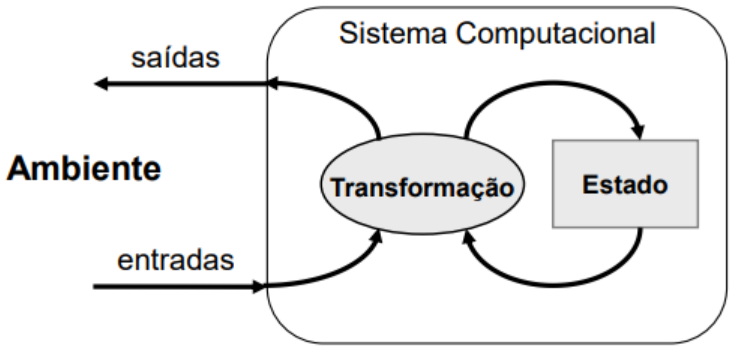

Figura 3.1: Modelo abstrato de um sistema computacional. Retirado de [**?**].

Os conceitos fundamentais de um sistema computacional são:

- **Dinâmica**: Descreve os estados que um sistema pode assumir e a forma como eles evoluem ao longo do tempo, determinando o comportamento do sistema;
- **Comportamento**: Expressa a estrutura de controlo do sistema, que corresponde à forma como o sistema age (gera as saídas) perante a informação proveniente do exterior (entradas) e do seu estado interno;

3.3.1 Modelo formal de computação

A função de transformação, mencionada anteriormente, pode ser decomposta em duas funções distintas (ver tabela 3.2), em que **Q** representa o conjunto de estados possíveis do sistema (regularmente designado por espaço de estados), Σ representa o conjunto de entradas possíveis e **Z** representa o conjunto de saídas possíveis.

Tabela 3.2: Funções de Transformação de um Sistema Computacional

Designação	**Expressão**	**Descrição**
Função de Transição (δ)	$\delta : Q \times \Sigma \rightarrow Q$	Define a relação entre o estado do sistema e as entradas que recebe, e o próximo estado.
Função de Saída (λ)	$\lambda : Q \times \Sigma \rightarrow Z$	Define a relação entre o estado do sistema e as entradas que recebe, e a saídas que produz.

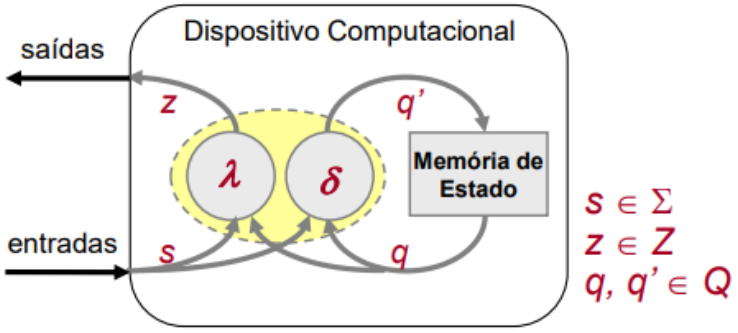

Figura 3.2: Modelo de um sistema computacional. Retirado de [**?**].

Associado a um determinado modelo de um sistema computacional, está uma máquina de estados, que representa, de forma abstrata, o comportamento do sistema em função do tempo. Pode ser caracterizada por:

- **Estado**: Se o número de estados possíveis é finito, então a máquina de estados é finita; caso contrário, é infinita;
- **Determinismo**: Uma máquina de estados finita também pode ser subcaracterizada em determinística (existe apenas uma transição para o mesmo estado e entrada) ou não determinística (existe mais do que uma transição para o mesmo estado e entrada). Para qualquer máquina de estados não determinística, existe uma máquina de estados determinística equivalente; [**?**, **?**]
- **Função de Saída**: Se a função de saída λ depende das entradas Σ, então a máquina de estados é do tipo *Mealy* ($\lambda : Q \times \Sigma \rightarrow Z$); caso contrário, é do tipo *Moore* ($\lambda : Q \rightarrow Z$).

3.4 Desenvolvimento do Jogo

Tendo por base os conceitos anteriormente abordados, que consolidaram a aprendizagem inerente à implementação da biblioteca, foi desenvolvido um jogo. No seu processo de desenvolvimento, foi definido um conjunto de componentes que providenciam implementações concretas dos subsistemas expostos pela biblioteca, adaptados ao contexto específico do jogo:

- **Personagem**: Representa o agente do jogo;
- **Ambiente**: Representa o ambiente onde a Personagem opera;
- **Máquina de Estados**: Representa a máquina de estados associada ao controlo da Personagem. Caracterizada por ser finita, não determinística e do tipo *Mealy* (Ver figura 3.3).

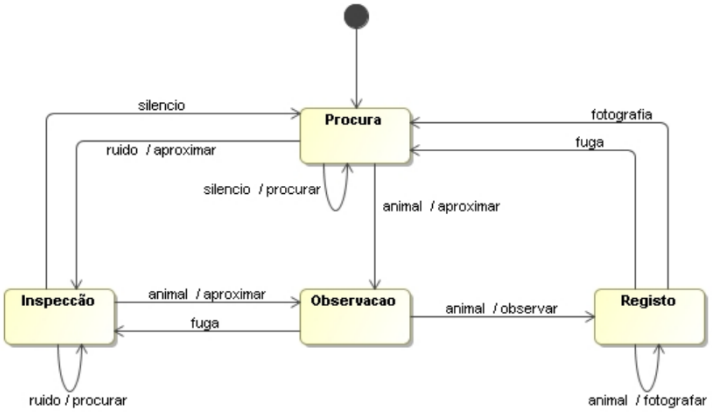

Figura 3.3: Máquina de estados associada ao controlo da Personagem. Retirado de [**?**].

O jogo consiste num ambiente onde a personagem tem por objectivo registar a presença de animais através de fotografias. O ambiente deste jogo, pode ser caracterizado (ver secção 2.2) por ser:

- **Virtual**: O ambiente é simulado e não existe fisicamente;
- **Totalmente Observável**: O agente tem acesso a uma descrição completa do ambiente e que lhe permite resolver o problema em questão sem necessitar de guardar estado interno;
- **De agente único**: Apenas existe um agente a atuar, neste caso, uma Personagem;
- **Determinístico**: O estado seguinte é unicamente determinado pelo estado actual e pela acção do agente;
- **Sequencial**: Pois existe uma linha de acontecimentos que se sucedem (e.g., não é possível inspeccionar, registar ou observar algo sem ser feita primeiro uma procura);
- **Estático**: Não se altera enquanto o agente está a tomar uma decisão;
- **Discreto**: O número de estados possíveis é finito.

Foi criada uma aplicação (ver figura 3.4) de cli (i.e., command-line interface) em Java, que possibilita a interação com o jogador por meio de comandos em texto. Tendo em conta que a interface gráfica não era o foco desta parte do projeto, foi emulada através de texto descritivo.

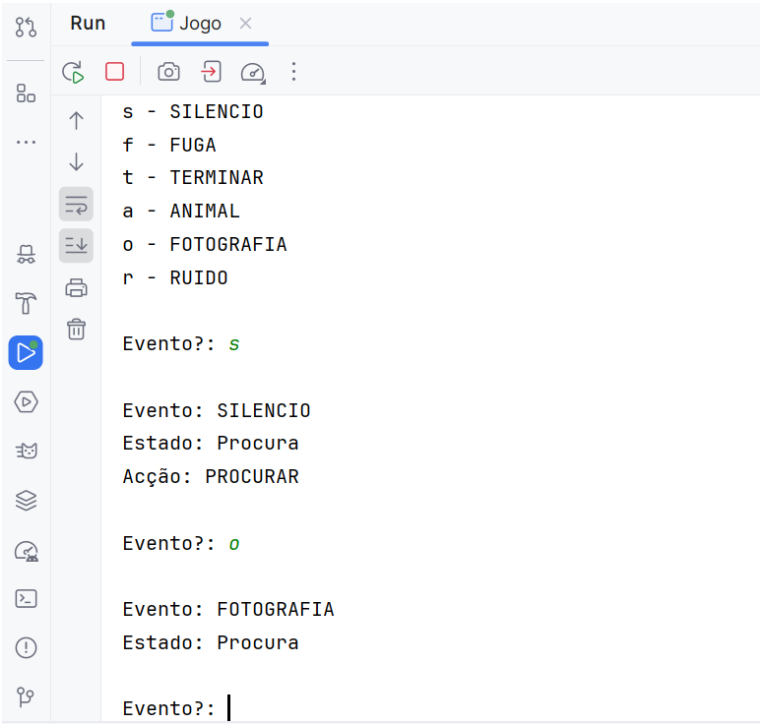

Figura 3.4: Utilização da aplicação do jogo.

3.5 Estrutura do Projeto

No processo de desenvolvimento de software associado a este projeto, foi definida a seguinte estrutura em módulos e que está presente na pasta iasa_jogo/src:

- *agente*: Integra classes e interfaces que definem o Agente e os seus componentes (e.g., módulo de controlo);
- *ambiente*: Agrega interfaces que representam o Ambiente e os seus componentes (e.g., comandos, eventos);
- *maqest*: Contém classes que definem o conceito de Máquina de Estados e os seus componentes (e.g., estados, transições);
- *jogo*: Agrega os detalhes da implementação do jogo, onde são integrados os módulos anteriores e definidas implementações concretas dos mesmos, adequadas ao contexto a que o jogo se insere.

Capítulo 4

Projecto - Parte 2

Este é o capítulo de testes. É possível forçar a inclusão de todas as referências com [].

Modo de matemática em texto $x = ma^2$ e em equação (duas formas):

$$x = ma^2$$

$$x = ma^2 \tag{4.1}$$

Capítulo 5

Revisão do Projeto

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Capítulo 6

Conclusão

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.