Logo

Description automatically generated

FACULDADE DE ENGENHARIAS E TECNOLOGIAS

Universidade Lusíada – *Campus* de Vila Nova de Famalicão

TECNOLOGIA LORA PARA O DESENVOLVIMENTO DE SISTEMAS IOT DE BAIXO CUSTO

Jaime Nunes Martins

Relatório Final de Desenvolvimento, realizado no âmbito da Unidade Curricular de Projeto de Engenharia Eletrónica e Informática

Maio 2023

FACULDADE DE ENGENHARIAS E TECNOLOGIAS

Universidade Lusíada – *Campus* de Vila Nova de Famalicão

TECNOLOGIA LORA PARA O DESENVOLVIMENTO DE SISTEMAS IOT DE BAIXO CUSTO

Jaime Nunes Martins

Relatório Intermédio de Desenvolvimento, realizado no âmbito da Unidade Curricular de Projeto de Engenharia Eletrónica e Informática

Orientador Interno: Vítor Pereira

# Agradecimentos

**Primeiramente gostaria de expressar a minha profunda gratidão à universidade e aos meus estimados professores, em especial ao meu orientador de projeto, o professor Vítor Pereira. Foi graças à sua orientação e dedicação que descobri a minha paixão pela programação. O Seu conhecimento e incentivo foram fundamentais para o meu crescimento acadêmico e pessoal.**

**Aproveito este momento para agradecer de todo o meu coração à minha querida Mãe, que sempre esteve presente, apoiando-me incondicionalmente. O teu apoio e motivação constantes têm sido a impulsão para superar os meus desafios.**

**Um agradecimento especial também ao meu irmão, cuja colaboração foi essencial na construção deste projeto. A sua paciência em me ajudar nos meus primeiros passos em Kotlin e Android, assim como o seu apoio incansável, foram de valor inestimável para o meu sucesso e o sucesso deste projeto.**

**Não posso deixar de mencionar a minha namorada, a quem agradeço pela ajuda na redação deste relatório, oferecendo ajuda em todos os momentos desta jornada. A tua presença tem sido uma fonte constante de força e coragem.**

**Por fim, um enorme agradecimento a todos os meus amigos e colegas. As vossas contribuições de amizade e apoio tornaram esta experiência ainda mais significativa e enriquecedora.**

**A todos vocês, um enorme obrigado. O vosso apoio e presença na minha vida fizeram toda a diferença, e sou extremamente grato por ter cada um de vocês na nesta jornada.**

# Resumo

Este trabalho explora a tecnologia LoRa (Long Range) para desenvolver um sistema IoT que adquire informação sobre as condições climáticas, com o uso de torres periféricas, de modo a monitorar a qualidade do ar e do solo através de sensores. Os elementos do sensor enviam informações por meio da tecnologia LoRa para uma torre central (torre mãe) em relação à área de monitoramento. Os dados são armazenados e processados numa base de dados e apresentados numa aplicação android desenvolvida com essa finalidade.

A pesquisa concentrou-se no desenvolvimento de uma aplicação android, que permitirá obter e analisar os dados recolhidos pelas torres periféricas em tempo real.

Concluiu-se que o uso da tecnologia LoRa pode ser extremamente eficiente para o desenvolvimento de sistemas IoT de baixo-custo e com benefícios\* (adicionar os benefícios).

**Palavras-Chave:** Sistemas IoT, Android, Eficiência, Baixo-custo

# Abstract

**One of the components of an IoT (Internet of Things) system is wireless networks. Finding the best solution for an IoT application becomes a challenge as a wireless network is dependent on the autonomy of power systems. This work explores an emerging LoRa (Long Range) technology, which allows for the creation of low-cost IoT systems, as this technology is independent of telecommunication operators. This work consists of developing an IoT system for acquiring information about weather conditions (peripheral towers for monitoring both air and soil quality). The various sensor elements send information through LoRa technology to a gateway located at a central point (mother tower) in relation to the monitoring area. This information is sent to a database where it is stored and processed for presentation in an application developed for this purpose. The study is focused on identifying suitable solutions in relation to the autonomy of the power system (batteries) and also regarding the optimized frequency for sending data from sensors using LoRa technology. For this purpose, a triangulation system was designed consisting of two peripheral towers and a mother tower. The peripheral towers transmit data about the soil temperature and humidity to the mother tower. Regarding the choice of batteries, it was found that the best result for a sending frequency of 6 sets of data per hour were achieved with LiPo 3.7 V 1350 mA batteries. Considering the sending of information, data from the two peripheral towers were sent to the mother tower at the frequency of 868MHz, respecting the maximum transmission power. However, a solution is being developed to solve a problem associated with synchronization between the peripheral towers and the mother tower. This is because, if there is simultaneous sending of information from the peripheral towers after re-initialising the sending of information, sometimes data loss is observed. In this respect, a 4% loss of data was observed in relation to the sending and receiving of data. Nevertheless, it can be concluded that the use of LoRa technology can be efficient for the development of low-cost IoT systems.**

**Keywords:** IoT systems, LoRa technology, Efficient and Low-Cost communication

# Índice

[Agradecimentos iii](#_Toc140544108)

[Resumo iv](#_Toc140544109)

[Abstract v](#_Toc140544110)

[Índice vi](#_Toc140544111)

[Lista de Figuras viii](#_Toc140544112)

[Lista de Abreviaturas e Símbolos ix](#_Toc140544113)

[1 Introdução 11](#_Toc140544114)

[1.1 Enquadramento 11](#_Toc140544115)

[1.2 Objetivos 12](#_Toc140544116)

[1.3 Metodologia 12](#_Toc140544117)

[1.4 Estrutura do Relatório 12](#_Toc140544118)

[2 Enquadramento Teórico 13](#_Toc140544119)

[2.1 O que é IoT 13](#_Toc140544120)

[2.2 O que é o LoRa 14](#_Toc140544121)

[2.3 IoT com tecnologia LoRa 14](#_Toc140544122)

[2.4 Engenharia de Software 15](#_Toc140544123)

[2.5 Cascata 15](#_Toc140544124)

[2.6 Desenvolvimento evolutivo 18](#_Toc140544125)

[2.7 Metodologia 19](#_Toc140544126)

[2.7.1 Requisitos e Desenho 19](#_Toc140544127)

[2.7.2 Implementação 20](#_Toc140544128)

[2.7.3 Validação 20](#_Toc140544129)

[3 Percurso Prático 21](#_Toc140544130)

[3.1 Desenvolvimento da Aplicação “WeatherTower” 21](#_Toc140544131)

[4 Resultados e discussão 25](#_Toc140544132)

[5 Conclusões e perspetivas de trabalho futuro 27](#_Toc140544133)

[Referências Bibliográficas 29](#_Toc140544134)

# Lista de Figuras

[Figura 1- Arquitetura da Internet das Coisas [6]. 13](#_Toc140583196)

[Figura 2- Funcionamento Sistema LoRa [10]. 14](#_Toc140583197)

[Figura 3- Metodologia de desenvolvimento de software: Cascata [16]. 16](#_Toc140583198)

[Figura 4-Cascata modificada [23]. 17](#_Toc140583199)

[Figura 5 - Modelo Evolutivo [26]. 18](#_Toc140583200)

[Figura 6 – Fluxograma resultado do desenho[33] 22](#_Toc140583201)

[Figura 7- Link da API com os dados selecionados. 23](#_Toc140583202)

[Figura 8 - Comparação de dados com a google 24](#_Toc140583203)

# Lista de Abreviaturas e Símbolos

|  |  |
| --- | --- |
| **Sigla** | **Descrição** |
| IoT | Internet of Things |
| LoRa | Long Range |
| PM | Particular Matter |
| GPS | Global Positioning System |
| IEE | European Environment Agency |
| APP | Aplicação |
|  |  |

# Introdução

A poluição do ar é um problema global que se tem agravado nas duas últimas décadas. De acordo com a EEA (European Environment Agency), verificou-se um decréscimo substancial na emissão de gases poluentes durante os últimos anos observando-se, consequentemente, uma melhoria da qualidade do ar na Europa [1]. Contudo, as concentrações de ar poluente continuam demasiado altas, principalmente nos centros urbanos onde é possível encontrar em excesso três poluentes reconhecidos pela EEA como os mais afetam a saúde humana, sendo este ozono (O3), dióxido de nitrogénio (NO2) e material particulado (“particulate matter” ou PM, na literatura anglo-saxónica).

Face a esta problemática, verifica-se uma coesão das políticas governamentais, a nível mundial, no combate às alterações climáticas. Contudo, em 2022 constatou-se a existência de países que ainda excedem o limite máximo de emissão de gases poluentes. Com base nesta evidência torna-se necessária a monitorização da qualidade do ar em todo o território mundial [2].

## Enquadramento

Com o aparecimento do conceito “Internet of Things” (IoT), que carateriza a interconexão de vários dispositivos em relação a sua acessibilidade e inteligência incorporada, também se iniciou uma nova era de monitoramento.

Sistemas de monitorização são geralmente usados para observar com regularidade e adquirir dados, e estes têm tido um papel importantíssimo nos dias de hoje. Bons sistemas de monitorização ajudam a notificar os seus utilizadores de anormalidades que possam estar a acontecer, levando assim a uma rápida contramedida face ao problema [3]. IoT é um conceito tecnológico que projeta um mundo onde objetos do dia-a-dia estão equipados com microcontroladores e dispositivos de comunicação de modo que máquinas possam comunicar entre si. Contudo, surgem assim alguns desafios como a eficiência da comunicação, o baixo consumo de energia, a distância a ser percorrida pelos pacotes de dados assim como a latência das comunicações. Para este propósito o LoRa é o candidato ideal para responder a estas necessidades com uma vasta aplicação “smart”, como cidades inteligentes ou “smart cities”, “smart healthcare”, indústria, ou neste caso monitorização ambiental. [4]

## Objetivos

Os sistemas convencionais projetados para monitorização têm algumas limitações em termos de falta de tecnologias suportadas via wireless, capacidade de processamento dos microcontroladores e acessibilidade da informação recolhida por estes sistemas.

Este estudo propõe uma relação custo eficiência baseado no LoRa e na IoT com uma interface amiga do utilizador. O objetivo é uma conexão de longo alcance com baixo consumo de energia através da interface rádio do LoRa, recolha de múltiplos parâmetros com recorrência a vários sensores numa só torre (“smart unit”), autossustentabilidade com recurso a energias renováveis e dados em tempo real através de uma aplicação android.

O foco deste projeto foi o desenvolvimento de uma aplicação que possa ser usada para receber qualquer tipo de dados, de acordo com a necessidade do utilizador e observá-los de uma forma fácil e intuitiva.

## Metodologia

Numa fase inicial do projeto foi feito um estudo das tecnologias inerentes aos sistemas da IoT, identificando os componentes que davam melhor resposta aos objetivos pretendidos neste projeto. Seguiu-se a seleção da linguagem de programação em que seria implementada a aplicação android. Esta seleção suscitou algumas dúvidas entre duas linguagens orientadas aos objetos, que seriam a base da implementação: Java ou Kotlin, sendo Kotlin o escolhido [5]. Após esta escolha foi então iniciado um estudo aplicado à engenharia de software, voltado para a especificação, desenvolvimento, manutenção e criação de software.

## Estrutura do Relatório

Este documento está divido em cinco capítulos com a seguinte estrutura:

* Capítulo 1: Introdução
* Capítulo 2: Enquadramento Teórico
* Capítulo 3: Percurso Prático
* Capítulo 4: Resultados e Discussão
* Capítulo 5: Conclusão.

# Enquadramento Teórico

## O que é IoT

A IoT (Figura 1) refere-se à interconexão de dispositivos físicos, veículos, edifícios, e outros objetos embebidos com eletrónica, software, sensores, e conexão à internet. Isto permite aos objetos recolher e trocar dados com outros dispositivos e sistemas, tornando-os inteligentes (“smart”) e capazes de poder tomar decisões autonomamente. A IoT tem o potencial de revolucionar muitos aspetos das nossas vidas, desde sistemas de saúde e transportes a produção industrial e agricultura.

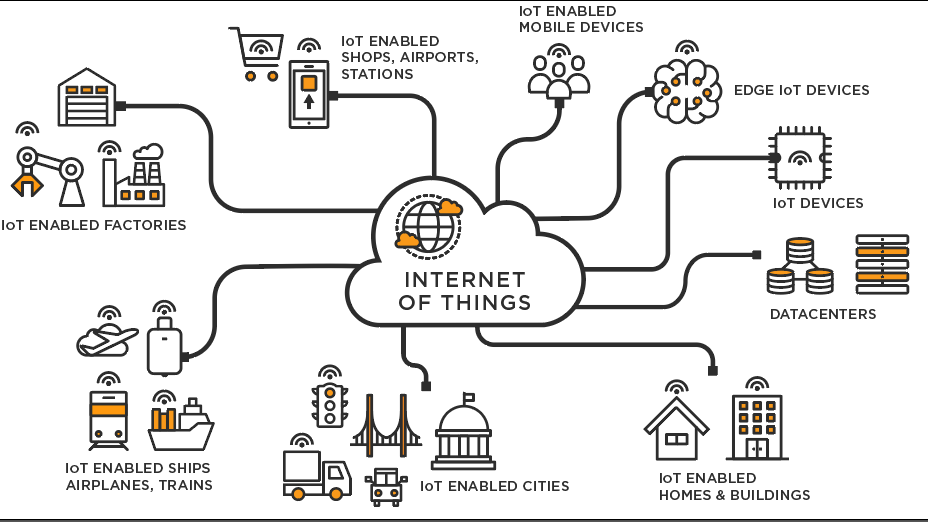


Figura 1- Arquitetura da Internet das Coisas [6].

O termo “Internet of Things” foi criado, oficialmente, nos anos 2000, apesar da ideia de dispositivos ligados já existir desde os anos 70 do século passado [7]. Nesse tempo eram usados os termos internet embebida (“embedded internet”) ou computação ubíqua (“pervasive computing”), mas o termo agora conhecido “IoT” foi apresentado por Kevin Ashton, em 1999 durante o seu trabalho na empresa Procter & Gamble [7]. Ashton que na época se encontrava a trabalhar com uma nova tecnologia chamada identificação por radiofrequência (“radio frequency identification” ou “RFID”) para conectar objetos através das RFID. A internet era a tendência em 1999 , o que influenciou Kevin Ashton a intitular a sua apresentação de “IoT” [7].

## O que é o LoRa

LoRa diminutivo do inglês de Longo Alcance (“Long Range”) é uma tecnologia de comunicação sem fios desenvolvida pela Semtech em 2012 [8]. O LoRa foi desenvolvido para providenciar comunicações de longo alcance, e de baixo consumo para dispositivos IoT e outras aplicações com necessidade de uma comunicação sem fios. O LoRa usa uma tecnologia de modelação de rádio frequência, convertendo sinais de dispositivos, como por exemplo, ondas de rádio ou de televisão como podemos ver na figura 2, para que possam ser processados e recebidos pelos respetivos dispositivos a longas distâncias, conseguindo alcançar até 10 km em alguns casos. É também conhecido pelos seus baixos consumos de energia, e pela capacidade de operar em ambientes desafiantes com altos níveis de interferência [9].

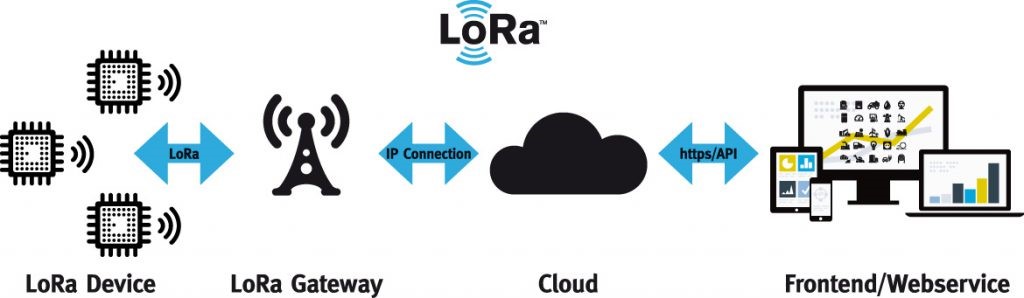


Figura 2- Funcionamento Sistema LoRa [10].

## IoT com tecnologia LoRa

O LoRa está a ser usado por uma ampla variedade de aplicações, desde agricultura inteligente a cidades inteligentes.

No âmbito da agricultura inteligente, a tecnologia LoRa está a ser usada para monitorar os níveis de água no solo, temperatura, humidade, e outros fatores ambientais críticos para o desenvolvimento da agricultura [11].

Esta tecnologia está a ser usada, também, nas cidades inteligentes para monitorar a qualidade do ar, trânsito nas estradas e a produção industrial.

Um exemplo da aplicação desta tecnologia é a utilização de um conjunto de sensores, espalhados pela cidade, com o objetivo de medir a qualidade do ar e detetar a presença de poluentes. Os dados recolhidos pelos sensores serão, posteriormente, analisados com o intuito de identificar as áreas com uma qualidade do ar abaixo do expectável, para que se possa intervir e melhorar a mesma [12].

O sistema LoRa pode, também, ser usado para detetar lugares de estacionamento vazios, facilitando os condutores a encontrar parque rapidamente o que por consequência reduz o congestionamento nas estradas [13].

Esta tecnologia, em ambiente de fábrica, é utilizada para monitorar várias linhas de produção em tempo real, acompanhar o progresso de cada fase de produção e identificar produtos defeituosos baseado no “delay” da produção. Podemos obter também com a ajuda deste sistema, alertas do mal funcionamento de alguma maquinha, ou necessidade de manutenção, obtendo assim uma rápida intervenção para evitar custos acrescidos devido ao mal funcionamento das mesmas. Para além disto, os dados recolhidos através dos dispositivos de IoT propulsionam oportunidades de melhoria, na eficiência e produtividade. [14]

## Engenharia de Software

A Engenharia de Software é uma área da engenharia da computação voltada para a especificação, desenvolvimento, manutenção e criação de software, com aplicação de tecnologias e práticas de organização de projetos e outras disciplinas, visando organização, produtividade e qualidade.[15] O desenvolvimento de software é considerado uma atividade tediosa e demorada, que requer grandes recursos financeiros e humanos, e, como tal há necessidade de preparar e planear adequadamente os projetos de software para que sejam concluídos com sucesso e dentro do prazo. Nos seguintes subcapítulos são apresentadas duas metodologias, cascata sendo o modelo tradicional e desenvolvimento evolutivo que foi o modelo seguido neste projeto.

## Modelo em Cascata

No modelo em cascata o desenvolvimento ocorre em ordem sequencial, ou seja, uma etapa deve ser concluída antes de se prosseguir para a próxima etapa e, portanto, para se fazer alterações, é necessário seguir um processo formal. Estas etapas são repetidas recursivamente até que o software esteja completamente desenvolvido. As etapas incluem a análise (especificação de requisitos), design, implementação, teste e manutenção. Isto pode ser ilustrado da seguinte forma:

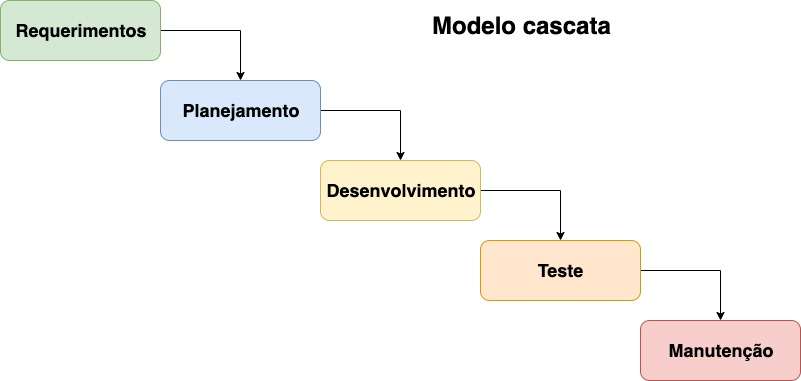


Figura 3- Metodologia de desenvolvimento de software: Cascata [16].

A abordagem tradicional do desenvolvimento de software pode ser ilustrada através do modelo em cascata. Este modelo representa um modelo estático e aborda o desenvolvimento de software de uma forma linear e sequencial, ou seja, concluindo uma etapa antes de outra se iniciar. Fowler [17] afirma que o desenvolvimento em cascata divide o projeto baseados nas suas atividades: análise de requisitos, planeamento, implementação de código, e teste. Pressman [18] identifica as atividades como: comunicação (envolve o início do projeto e recolha de requisitos), planeamento (estimativa de custo, programação e acompanhamento), modelagem (análise e planeamento), construção (implementação de código e teste) e lançamento (entrega, suporte e feedback). Pfleeger e Atlee [19] apresentam o modelo como envolvendo as seguintes fases: análise de requisitos, planeamento do sistema, planeamento de programa, implementação de código, teste e integração, teste de sistema, teste de aceitação, operação e manutenção. Resumidamente, o modelo em cascata pode ser dito que envolve as seguintes fases: análise de requisitos, planeamento, implementação, teste e manutenção [20].

O modelo em cascata geralmente tem metas distintas para cada fase de desenvolvimento do projeto. Uma vez que uma fase é completamente desenvolvida, o projeto avança para a próxima fase não existindo a oportunidade de voltar e revisitar estágios anteriores, conforme ilustrado na figura 3. Cada fase concluída alcança uma meta e um objetivo específico, que por norma é bastante diferente do objetivo da próxima fase. Além disso, durante o planeamento, se um erro for detetado nas fases já concluídas, geralmente não há oportunidade de revisitar a fase anterior. Por exemplo, durante a fase de planeamento, pode surgir algo que exija revisitar a fase de requisitos. Deve-se observar que, durante o processo de desenvolvimento, pode ser necessário fazer uma alteração devido a um ajuste na especificação de requisitos proposto pelo cliente. Tal alteração é impossível de ser alcançada no processo de desenvolvimento em cascata, o que demonstra a fraqueza da abordagem tradicional [21].

Este modelo é extremamente útil no desenvolvimento de sistemas estruturados, onde a alteração do software após a implementação do código é proibida. Além disso, processos e tratamento de dados são geralmente separados no modelo em cascata, de modo que, se os dados forem modificados, o código também deve ser alterado. Isto torna o software não reutilizável e muito difícil de atualizar, pois todos os processos precisam de ser modificados para qualquer ajuste, o que se torna trabalhoso e dispendioso .

Recentemente, existiram algumas melhorias no modelo cascata com o intuito de abordar os problemas inerentes ao modelo tradicional. Estas melhorias resultaram nos modelos de “Desenvolvimento Rápido” que McConnell [22] intitula de “cascatas modificadas". Ao contrário do modelo tradicional, o modelo modificado permite que as fases dos projetos se sobreponham mantendo as fases do modelo tradicional: análise de requisitos, planeamento, implementação, teste e manutenção, mas neste cada etapa depende da seguinte e da anterior, respetivamente (figura 4).

Uma imagem com texto, captura de ecrã, diagrama, file

Descrição gerada automaticamente

Figura 4-Cascata modificada [23].

A sobreposição de fases proporciona flexibilidade no processo de engenharia de software. Assim, podemos garantir que os defeitos no sistema de software são removidos na fase de desenvolvimento, reduzindo assim o custo adicional em alterações antes da fase de implementação. Apesar da sobreposição de fases, um projeto de software baseado no modelo cascata modificada ainda está sujeito a atrasos devido à dependência de uma fase em relação à anterior. No entanto, essa desvantagem pode ser eliminada estabelecendo-se um ponto de referência antes do início do projeto. Como resultado, muitos sistemas de informação e projetos adotaram o modelo modificado [24].

## Desenvolvimento evolutivo

Desenvolvimento evolutivo é mais um dos muitos modelos da engenharia de software, na qual o software deve ser desenvolvido de forma a evoluir a partir de protótipos iniciais. Para compreender melhor este modelo, é essencial entender o conceito de prototipagem. A prototipagem é uma abordagem que se baseia numa perspetiva evolutiva do desenvolvimento de software, afetando o processo na sua totalidade. Esta abordagem implica a criação de versões iniciais – “protótipos” - de um sistema futuro, por meio dos quais é possível realizar verificações e experiências, de modo a avaliar algumas das suas caraterísticas [25].

Num projeto de software inúmeras questões podem ser respondidas com a construção de protótipos, oferecendo suporte e orientação ao cliente quando este não tem uma ideia exata dos requisitos da nossa aplicação. Este modelo pode ser ilustrado da seguinte forma:Uma imagem com texto, captura de ecrã, diagrama, Tipo de letra

Descrição gerada automaticamente

Figura 5 - Modelo Evolutivo [26].

No modelo evolutivo, os requisitos são levantados de modo paralelo à evolução da aplicação à medida que vão sendo desenvolvidos protótipos, que ajudam a perceber os verdadeiros requisitos do sistema, obrigando a uma comunicação constante e direta com o cliente. Na figura 5 podemos notar que “versões intermediárias” se carateriza como um bloco, isto porque sempre que é recebido feedback, é iniciado um novo projeto de desenvolvimento. Isto é, a cada nova versão do software, melhorias são aplicadas para garantir a maior satisfação do cliente. No entanto, o facto de poderem existir inúmeras versões do sistema, pode resultar numa fraca estruturação do mesmo, devido à falta de visibilidade do processo. Porém, a vantagem deste modelo, é a possibilidade de alcançar uma elevada satisfação por parte do cliente. Como ele realiza feedbacks contínuos, as chances de um requisito não estar implementado, são menores [25].

## Metodologia

### Requisitos e Desenho

Na engenharia de software, a análise de requisitos abrange todas as tarefas relacionadas com a investigação e definição de novos sistemas. A análise de requisitos desempenha um papel crucial no processo de desenvolvimento de software, no qual o cliente e a equipa designada identificam as necessidades do projeto. Uma vez identificados os requisitos do sistema, a equipa encontra-se apta para desenvolver a solução.

A análise de requisitos é uma das primeiras etapas do desenvolvimento de software. O seu resultado é a especificação de necessidades, que define o software em duas dimensões: requisitos funcionais e requisitos não funcionais. Nesta fase, são realizadas a primeiras reuniões com os clientes ou utilizadores do software para compreender as funcionalidades do sistema a serem desenvolvidas. É também nesta fase que ocorrem a maioria dos erros, uma vez que a falta de experiência dos clientes nem sempre lhes permite ter clareza sobre quais funcionalidades o software terá [27].

A fase de desenho, no contexto da engenharia de software, é o processo de planeamento, criação de estrutura e arquitetura do software. Durante este processo, todos os requisitos e especificações são traduzidos para um desenho detalhado da aplicação.

O desenho de software envolve a definição das componentes do sistema, a sua iteração e a forma como as funcionalidades serão implementadas. Isso inclui a seleção de padrões de desenho adequados, a definição de classes, módulos e interfaces, e a organização geral do código.

Durante o desenho, é importante considerar a escalabilidade, a manutenibilidade e a extensibilidade do software, de forma a garantir que ele possa ser facilmente modificado ou expandido no futuro. O desenho deve levar em conta os requisitos não funcionais, como desempenho, segurança e usabilidade, com o fim de garantir que o software atende às expectativas dos utilizadores [28].

### Implementação

A implementação, é a etapa em que o código-fonte é escrito e o software começa a ser construído. Nesta fase o programador, utiliza as especificações e o design do software para desenvolver as funcionalidades planeadas.

Durante a implementação, é importante seguir as melhores práticas de codificação e utilizar as linguagens de programação adequadas ao projeto. O programador deve garantir a modularidade do código, para facilitar a manutenção futura e permitir a reutilização de partes do software. Além disso, é essencial realizar testes durante a implementação para verificar se o software está a funcionar conforme o esperado, e se está livre de erros. Durante a implementação é comum o uso de ambientes de desenvolvimento integrado e ferramentas de controle de versão, que auxiliam no desenvolvimento colaborativo e no controle das alterações feitas no código [29].

A implementação é uma etapa fundamental do ciclo de vida do desenvolvimento de software e representa a concretização das ideias e dos planos definidos anteriormente.

### Validação

No contexto da engenharia de software, a validação é o processo de avaliação e verificação do software que garante que este atende aos requisitos e às expectativas dos utilizadores. É na validação que são realizados testes e verificações para identificar possíveis erros, falhas ou discrepâncias entre o software desenvolvido e as especificações definidas. Isto inclui a validação das funcionalidades, desempenho, usabilidade e outras características relevantes do software.

Os testes de validação podem abranger diferentes níveis, desde testes unitários para verificar o correto funcionamento de partes isoladas do software, até testes de integração para garantir a interação adequada entre os diferentes módulos e componentes.

A validação também pode envolver a participação dos utilizadores finais, através de teste de aceitação, para verificar se o software atende às suas necessidades e expectativas, assegurando a qualidade, confiabilidade e adequação do software em relação aos requisitos estabelecidos [29].

# Percurso Prático

## Desenvolvimento da Aplicação “WeatherTower”

A aplicação (APP) “WeatherTower” foi desenvolvida segundo as práticas da engenharia de software, adotando uma abordagem evolutiva no seu desenvolvimento.

Numa fase inicial, o foco foi adquirir conhecimentos em linguagens de programação relevantes, como Java e Kotlin através da leitura de livros especializados em desenvolvimento Android, nomeadamente, “Head First Kotlin: A Brain-Friendly Guide” [30], “Head First Android Development” [31] e “Android Programming: The Big Nerd Ranch Guide” [32].

A metodologia utilizada na construção da aplicação “WeatherTower” seguiu os passos da engenharia de software. Iniciou-se com a análise de requisitos, onde foram identificadas as necessidades e funcionalidades esperadas da aplicação, sendo a mais importante a análise em tempo real dos dados recebidos, através do LoRa integrado na torre periférica. Juntamente com a receção de dados, outro aspeto importante seria a mostragem destes dados, e a utilização do “Global Positioning System” (GPS), para localizar a torre periférica mais próxima, assim os dados são recebidos com maior precisão possível para a localização atual. Para finalizar, foi pensado um sistema de armazenamento (repositório) para que no futuro seja fácil a implementação de um sistema de cache para armazenamento de dados. Para programação do código, utilizou-se o Kotlin por ser a linguagem mais atualizada para a programação android.

Após a análise de requisitos, deu-se início à fase de desenho da APP. Nesta etapa, a estrutura e a arquitetura do software foram planeadas. Primeiramente, optou-se por se utilizar a API *open-meteo*. Levando em consideração as informações essenciais para o funcionamento da APP, foi selecionado uma série de dados a serem recebidos pelo aplicativo.

Em paralelo, foram escolhidas as melhores dependências para trabalhar com esses dados. Nesse sentido, as bibliotecas empregues foram: a Retrofit para a comunicação com a API, o Serviço de Localização para obter dados relevantes e o Dagger-Hilt para a injeção de dependências.

Além disso, durante a fase de desenho, foi dedicado tempo para planear e projetar a interface do utilizador (UI) de forma intuitiva e agradável. O objetivo foi criar uma experiência de utilização fluida de forma a facilitar a interação dos utilizadores com a APP.

Uma imagem com texto, Nota em post-it, captura de ecrã, diagrama

Descrição gerada automaticamente

Figura 6 – Fluxograma resultado do desenho[33]

Estas decisões foram fundamentais para estabelecer as bases do projeto e garantir que a implementação seguisse as melhores práticas da engenharia de software.

Com o desenho definido, prosseguiu-se para a implementação. O desenvolvimento da APP ocorreu na ferramenta *Android Studio*,versão 4.2.2, uma ferramenta que possibilita a criação e o desenvolvimento de aplicativos Android de forma eficiente e produtiva.

## Importação dos dados

Numa primeira fase do desenvolvimento, foram selecionados os dados considerados mais relevantes (figura 7) tendo em consideração a Torre Periférica já construída em laboratório. Num leque abrangente de dados, os selecionados para a APP foram: a hora de leitura, a temperatura a 2 metros do solo (2m), humidade relativa 2m, velocidade do vento 10m, humidade no solo entre 1 e 3 cm de profundidade e o código meteorológico. Este código é o ajuda a nossa aplicação a decidir a imagem que melhor se adequa ao tempo atual e à previsão do mesmo.

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura 7- Link da API com os dados selecionados.

Estes dados são recebidos em formato JSON, e convertidos na nossa APP para poderem ser lidos corretamente. Este processo de receber e converter os dados ocorre numa co-rotina (“suspend function”) pois pode demorar muito tempo dependendo da quantidade de dados importados, e assim esta atividade é efetuada sem interromper as atividades principais. A API selecionada retorna uma coleção de dados no formato JSON, que contém informações relevantes para a minha aplicação. No entanto, para os poder utilizar de forma eficiente, é necessário realizar a conversão desses dados para objetos que sigam a estrutura definida na minha classe de dados (figura 8).

Para isso faremos uso dos “data transfer objects” (DTO), que atuam como intermediários entre a resposta da API e a representação dos dados na minha aplicação. Os DTOs permitem mapear os campos e propriedades do ficheiro JSON para os atributos da classe de dados, os mesmo ficariam armazenados em listas, facilitando a manipulação e o processamento dessas informações.

A conversão de dados envolve extrairos valores relevantes do JSON e atribuí-los aos atributos correspondentes no meu ficheiro DTO. Em seguida, podemos usar os objetos DTO para criar instâncias da minha classe de dados, garantindo que estejam formatadas corretamente de acordo com as necessidades da aplicação.

Esta abordagem de utilizar DTOs é geralmente adotada para separar a parte de transporte dos dados, da parte de implementação dos objetos criados por estes dados, proporcionando uma estrutura flexível e consistente para o processamento dos dados obtidos pela API.

Uma imagem com texto, captura de ecrã, software, ecrã

Descrição gerada automaticamente

Figura 8 - Classe de dados

## Localização

A preocupação seguinte foi integrar um sistema de localização, para obtenção de dados locais e atualizados, com a ajuda da livraria “play-service-location”. Esta biblioteca esta dependente de acesso a interner e GPS, mas sabendo que o GPS por sua vez está dependente de internet, apenas uma verificação foi necessária. Foi criada uma classe “DefaultLocationTracker” que é uma implementação da interface “LocationTracker” anotada como uma co-rotina,, de modo que não interrompa a linha principal da aplicação. A classe implementa uma função suspensa, (suspensa pois estamos a trabalhar com co-rotinas) esta função com o nome, “ getCurrentLocation()” tem como objetivo obter a localização atual, e retornar um objeto do tipo “Location”. O “?” na figura 9 depois do tipo “Location” indica que o retorno pode ser nulo, ou seja, pode ser retornado um valor “Location” ou “null”.

A função “getCurrentLocation()” (figura 9) definida na classe “DefaultLocation Tracker”, é uma função suspensa que tem função obter a localização atual. Em primeiro lugar, são verificadas as permissões de localização, tanto a localização precisa (“”Manifest.permission.ACESS\_FINE\_LOCATION”) como a permissão de localização aproximada (“Manifest.permission.ACESS\_COARSE\_LOCATION”). Se alguma dessas permissões não tiver sido concedida, a função retorna “null”, indicando que não é possível obter a localização atual.

Imediatamente depois, é verificado se o GPS está ativado. Isso é feito utilizando o “LocationManager” para verificar a localização via (“LocationManager.NETWORK\_PROVIDER”), dando uma resposta verdadeira ou falso em “boolean” para a minha variável “isGpsEnabled”. Se o GPS estiver desativado, a função também retorna “null” indicando que não é possível obter a localização atual.

Caso todas as verificações sejam bem sucedidas, a função utiliza o uma ferramenta da biblioteca chamada “FusedLocationProviderClient” para obter a última localização conhecida, de modo a poupar recursos. Se a localização estiver disponível imediatamente, ela é retornada diretamente. Caso contrário, a função aguarda a disponibilidade da localização atual usando uma “suspendCancellableCoroutine”. Em caso de sucesso, a localização atual é retornada pela função. Se ocorrer um erro ao obter a localização a função retorna “null”. Álem disto, se a chamada for cancelada, a função também é cancelada.

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Figura 9 - "Função getCurrentLocation"

## Tratamento dos dados

Com a obtenção da localização, restou apenas o tratamento de dados e a interface gráfico. No código foram utilizados mapas “chave-valor” para converter e organizar os dados de forma adequada.

O método “toWeatherDataMap()” desempenha um papel fundamental ao converter os dados recebidos. Ele percorre a lista de horas, e para cada hora, mapeia os dados correspondentes como temperatura, código climático, velocidade do vento, humidade no solo e humidade relativa. Esses dados são então guardados em objetos da classe “WeatherData” (figura 10) em que a hora é convertida para o formato apropriado usando “LocalDateTime.parse()”, sendo que o “parse()” é o método que desempenha o papel de analisar a “string” (sequência de caracteres) e criar um objeto.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura 10 - Classe de dados para criação dos objetos

Posteriormente, os dados são agrupados com base no índice, dividindo-os por 24 para criar grupos que representam cada dia (figura 11). O resultado é um mapa em que cada chave representa um dia e os valores associados são as listas de “WeatherData” correspondentes a esse dia. No método “toWeatherInfo()”, o mapa dos dados climáticos é obtido através do chamamento do método “toWeatherDataMap()”. Em seguida, é obtida a hora atual utilizando “LocalDateTime.now()”. O código verifica se há dados climáticos para o dia atual e encontra o objeto “WeatherData” correspondente à hora atual ou à hora seguinte, considerando que os minutos atuais são inferiores a 30 minutos. Essas informações são incluídas num objeto “WeatherInfo” que contém o mapa dos dados climáticos por dia e o objeto “WeatherData” correspondente ao clima atual.

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Figura 11 - Ficheiro onde é feito o mapeamento dos dados.

Desta forma, os dados climáticos são tratados e organizados de maneira apropriada, permitindo o seu uso na interface gráfica. A classe “WeatherInfo” é uma data class que contém as informações de dados climáticos por dia e o clima atual.

Resta apenas a implementação da interface gráfica, que recebe os objetos, e que os mostra no ecrã. O objetivo nesta fase, foi uma interação intuitiva e agradável para o utilizador de modo a criar uma experiência e utilização fluida.

Foi criada uma função chamada “WeatherCard”, que é responsável por exibir um “cartão” com as informações climáticas atuais. Ao receber o estado “WeatherState” (figura 12) como parâmetro, é efetuada a verificação para descobrir se “WeatherInfo” e currentWeatherData” disponíveis. Se sim, o desenho do cartão é criado utilizando a função “Card”. Este cartão possui um fundo de cor azul obtido com o código de cores “addressable RGB” (ARGB) e bordas redondas, através da função “RoundedCornerShape()” que recebe um único parâmetro de medida.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura 12 - Classe de dados para saber o estado corrente

Dentro do cartão, as informações climáticas são organizadas em coluna utilizando a função “Column”. Inicialmente, é exibida a hora atual no formato “HH:mm” (horas:minutos) e descendo um pouco, podemos encontrar a imagem correspondente ao nosso código metereológico, que é exibida utilizando a função “Image”, onde a imagem e injetada através do “painterResource”, que se trata de uma função da libraria “JetpackCompose” usado para carregar recursos de imagem, para a componente visual. Ao chamar “painterResource(id = R.drawable.ic\_weather)”, por exemplo, o “painterResource” carrega a imagem com a identificação “R.Drawable.ic\_weather” e retorna um objeto “Painter” correspondente a essa imagem. Esse objeto Painter contém os dados necessários para a renderizar, como o formato, a dimensão e os bytes da imagem.

As imagens foram retiradas em formato da “World Meteorological Organization” em formato “Scalable Vector Graphics” (SVG) para “Extensible Markup Language” (XML) (figura 13).

A temperatura atual é exibida em texto, utilizando a função “Text” com o valor de “data.temperatureCelsius” seguido do símbolo “ºC”.

Asseguir são exibidos os dados adicionais, como a humidade relativa, humidade no solo e velocidade do vento. Estes dados são organizados em linha na horizontal utilizando a função “Row”. Para cada dado a função “WeatherDataDisplay” é utilizada para criar a componente visual que exibe o valor, o ícone correspondente e a unidade de medida.

Dessa forma, a função “WeatherCard” é responsável por montar o cartão que exibe as informações climáticas atuais, proporcionando uma interface gráfica intuitiva e agradável ao utilizador.

Uma imagem com texto, captura de ecrã, software, Sistema operativo

Descrição gerada automaticamente

Figura 13 - Comparação de dados com a google

# Resultados e discussão

Ao longo do desenvolvimento da aplicação “WeatherTower”, foram alcançados resultados significativos que demonstram eficácia das metodologias de engenharia de software e implementação.

Em termos de conhecimento adquirido, a leitura de livros especializados em desenvolvimento Android, como “Head First Kotlin: A Brain-Friendly Guide” [30], “Head First Android Development” [31] e “Android Programming: The Big Nerd Ranch Guide” [32], foi fundamental para a compreensão das linguagens de programação relevantes, como Java e Kotlin. Essa base sólida permitiu o desenvolvimento da aplicação com maior confiança e competência.

A metodologia seguida durante a construção APP foi baseada no método de desenvolvimento evolutivo com vários protótipos de teste, e cada um deles, nos aproximávamos mais do objetivo. A análise em tempo real dos dados foi considerada a funcionalidade mais importante, sendo que para adaptar à Torre Periférica pequenas alterações seriam necessárias.

No que diz respeito à interface gráfica esta foi planeada para apresentar os dados recebidos de uma forma clara e compreensiva.

Em suma, os resultados obtidos com a aplicação “WeatherTower” demonstraram ser desafiantes desda aprendizagem de uma nova linguagem, à aplicação da metodologia de engenharia de software.

# Conclusões e perspetivas de trabalho futuro

Para trabalho futuro ficam em aberto a expansão das funcionalidades da APP. Isto inclui previsões de tempo de curto e longo prazo, gráficos históricos de dados meteorológicos entre outros. Estas melhorias na utilidade proporcionarão uma experiência mais abrangente ao utilizador.

Atualmente a APP utiliza a API “open-meteo” como fonte de dados meteorológicos. No entanto, uma prespetiva interessante seria explorar a possibilidade de integrar com a API da Torre Periférica.

Em conclusão, a aplicação “WeatherTower” demonstrou o potencial das boas práticas de engenharia de software e a escolha adequada de tecnologias e ferramentas. O sucesso do projeto está diretamente relacionado à análise de requisitos, ao planeamento cuidadoso, à seleção de dados relevantes, ao tratamento eficiente dos dados e ao design intuitivo da interface gráfica. A aplicação representa um avanço no acesso e na visualização de informações meteorológicas precisas e atualizadas.

# Referências Bibliográficas

[1] (<https://www.eea.europa.eu/themes/air/country-fact-sheets/2021-country-fact-sheets/portugal>)

[2] Lelieveld, J., Pozzer, A., Pöschl, U., Fnais, M., Haines, A., & Münzel, T. (2020). Loss of life expectancy from air pollution compared to other risk factors: a worldwide perspective. *Cardiovascular research*, *116*(11), 1910-1917.

[3] F. Akhter, H. Siddiquei, M.E.E. Alahi, S. Mukhopadhyay, Design and Development of an IoT-enabled Portable Phosphate Detection System in Water for Smart Agriculture, Sensors and Actuators A: Physical (2021), 112861.

[4] Ali, A. I., Partal, S. Z., Kepke, S., & Partal, H. P. (2019, June). ZigBee and LoRa based wireless sensors for smart environment and IoT applications. In *2019 1st Global Power, Energy and Communication Conference (GPECOM)* (pp. 19-23). IEEE.

[5] Bose, S., Mukherjee, M., Kundu, A., & Banerjee, M. (2018). A comparative study: java vs kotlin programming in android application development. *International Journal of Advanced Research in Computer Science*, *9*(3), 41-45.

[6] O que é a internet das coisas (IoT)?, https://www.tibco.com/sites/tibco/files/mediaentity/2020-05/IoT.png

[7] Gokhale, P., Bhat, O., & Bhat, S. (2018). Introduction to IOT. *International Advanced Research Journal in Science, Engineering and Technology*, *5*(1), 41-44.

[8] Olivier Bernard André Seller. (2014). Wireless communication method (US Patent No. 9647718B2). U.S. Patente. <https://patents.google.com/patent/US9647718B2/fr>

[9] Semtech, “*What are LoRa and LoRaWAN?*”, <https://loradevelopers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/> , visitado a 30/04/2023.

[10] Imagem encontrada em: [https://www.everythingrf.com/community/what-is- lora visitado a 30/04/2023](https://www.everythingrf.com/community/what-is-%20lora%20visitado%20a%2030/04/2023)

[11] S. J. Suji Prasad, M. Thangatamilan, M. Suresh, Hitesh Panchal, Christober Asir Rajan, C. Sagana, B. Gunapriya, Aditi Sharma, Tusharkumar Panchal & Kishor Kumar Sadasivuni (2022) An efficient LoRa-based smart agriculture management and monitoring system using wireless sensor networks, International Journal of Ambient Energy, 43:1, 5447-5450, DOI: [10.1080/01430750.2021.1953591](https://doi.org/10.1080/01430750.2021.1953591)

[12] Andrade, R.O.; Yoo, S.G. A Comprehensive Study of the Use of LoRa in the Development of Smart Cities. Appl. Sci. **2019**, 9, 4753. <https://doi.org/10.3390/app9224753>

[13] Sarker, V. K., Gia, T. N., Ben Dhaou, I., & Westerlund, T. (2020). Smart parking system with dynamic pricing, edge-cloud computing and lora. *Sensors*, *20*(17), 4669.

[14] Simo, A., Barbulescu, C., Kilyeni, S., & Dragos, C. (2018, May). LoRa based energy efficiency improving in manufacturing processes. In *2018 7th International Conference on Computers Communications and Control (ICCCC)* (pp. 156-161). IEEE.

[15] Schach, S. R. (1990). *Software engineering*. Aksen associates.

[16] Imagem retirada de: [https://pt.wikipedia.org/wiki/Modelo\_em\_cascata#/media/Ficheiro:Modelo\_em\_cascata.png](https://pt.wikipedia.org/wiki/Modelo_em_cascata" \l "/media/Ficheiro:Modelo_em_cascata.png)

[17] Fowler, Martin. *UML distilled: a brief guide to the standard object modeling language*. Addison-Wesley Professional, 2004.

[18] Fowler, Martin. *UML distilled: a brief guide to the standard object modeling language*. Addison-Wesley Professional, 2004.

[19] Pressman, Roger S. *Software engineering: a practitioner's approach*. Palgrave macmillan, 2005.

[20] Adenowo, A. A., & Adenowo, B. A. (2013). Software engineering methodologies: a review of the waterfall model and object-oriented approach. *International Journal of Scientific & Engineering Research*, *4*(7), 427-434.

[21] Munassar, N. M. A., & Govardhan, A. (2010). A comparison between five models of software engineering. *International Journal of Computer Science Issues (IJCSI)*, *7*(5), 94.

[22] McConnell, S.M. (1996). Rapid Development: Taming Wild Software Schedules. Microsoft Press.

[23] Martins, Jaime. “Cascata Modificada”.

[24] Model, W. (2015). Waterfall model. *Luettavissa: http://www. waterfall-model. com/. Luettu*, *3*.

[25] Batista, J., & Figueiredo, A. (1997). Desenvolvimento de programas educativos por prototipagem continuadamente evolutiva. *Actas do 2º Simpósio Investigação e Desenvolvimento de Software Educativo, Coimbra*

[26] Martins, Jaime. “Desenvolvimento Evolutivo”.

[27] Lethbridge, T. C., & Laganiere, R. (2005). *Object-oriented software engineering* (Vol. 11). New York: McGraw-Hill.

[28] Adnan, N. H., & Ritzhaupt, A. D. (2018). Software engineering design principles applied to instructional design: What can we learn from our sister discipline?. *TechTrends*, *62*, 77-94.

[29] Jabangwe, R., Edison, H., & Duc, A. N. (2018). Software engineering process models for mobile app development: A systematic literature review. *Journal of Systems and Software*, *145*, 98-111.

[30] Griffiths, D., & Griffiths, D. (2019). *Head first Kotlin: A brain-friendly guide*. O'Reilly Media.

[31] Griffiths, D., & Griffiths, D. (2017). *Head first android development: a brain-friendly guide*. " O'Reilly Media, Inc.".

[32] Hardy, B., & Phillips, B. (2013). *Android programming: the big nerd ranch guide*. Addison-Wesley Professional.