Resumo da parte do Guião sobre o projeto

O projeto envolve a criação de uma solução empresarial de múltiplas camadas que inclua a geração e processamento de fluxos de dados remotos, armazenamento persistente de dados e o desenvolvimento de uma API de serviço e um portal web para o acompanhamento dos dados. Aqui estão os pontos principais e exemplos concretos de projetos que podem ser desenvolvidos:

**Objetivos do Projeto:**

1. **Desenvolver uma especificação de produto** com base em cenários de uso (user stories).
2. **Propor e implementar uma arquitetura de software** utilizando frameworks empresariais.
3. **Aplicar práticas colaborativas de desenvolvimento** de código e gestão de projeto ágil.

Um exemplo de projeto sugerido no documento é um **sistema de gestão de parques eólicos**:

* **Conceito**: Uma plataforma para monitorar e controlar turbinas eólicas distribuídas em diferentes locais.
* **Aquisição de dados**: A telemetria das turbinas (rpm, geração de energia) e parâmetros ambientais (velocidade e direção do vento) são continuamente monitorados.
* **Publicação de dados**: Um gateway em cada parque agrega localmente os dados e os envia para a nuvem, utilizando um protocolo de comunicação eficiente.
* **Processamento**: Detecção de condições de perigo e envio de alarmes para dispositivos móveis.
* **API de integração**: Exposição de endpoints para que outros sistemas possam consultar informações das turbinas e dados de telemetria.
* **Portal Web**: Interface para monitoramento e controle das turbinas.

**Ferramentas e Arquitetura Recomendadas:**

* **Arquitetura Multicamadas**: Inclui camadas de fonte de dados, filas de mensagens, API (REST), lógica de negócio, e camada de persistência de dados.
* **Tecnologias e Ferramentas**:
  + **Backend**: Utilização de Spring Boot (Java) para a lógica de negócio e a API RESTful.
  + **Banco de dados**: PostgreSQL para persistência de dados históricos ou, dependendo do caso, uma solução NoSQL como MongoDB para escalabilidade.
  + **Processamento de Dados**: Kafka para filas de mensagens e processamento de fluxos de dados.
  + **Frontend**: React ou Angular para o desenvolvimento da interface web; Flutter ou React Native para o desenvolvimento de um aplicativo móvel.
  + **Contêineres e Deploy**: Docker para a criação de contêineres e Kubernetes para orquestração em um ambiente de cloud, como AWS ou Google Cloud.
  + **APIs e Documentação**: OpenAPI/Swagger para documentação da API.

Lista de APIs e Projetos Associados (com Dados Contínuos):

1. **Spotify Web API (usando o endpoint de Currently Playing Track)**
   * **Tema do Projeto**: Sistema de Monitorização Contínua de Música Escutada em Tempo Real.
     + Atualiza continuamente a música que o utilizador está a ouvir, permitindo analisar padrões de escuta ao longo do tempo e sugerir recomendações em tempo real.
2. **Google Fit API**
   * **Tema do Projeto**: Sistema de Monitorização Contínua de Saúde e Bem-Estar.
     + Recolhe dados de saúde, como batimentos cardíacos, passos, e outras métricas, continuamente a partir de dispositivos wearables e smartphones.
3. **Apple HealthKit API**
   * **Tema do Projeto**: Plataforma de Acompanhamento Contínuo de Sinais Vitais para Monitorização de Pacientes.
     + Obtenção contínua de dados de batimentos cardíacos, saturação de oxigênio e outras métricas, alertando para possíveis condições críticas.
4. **OpenWeatherMap API (usando dados de clima em tempo real)**
   * **Tema do Projeto**: Sistema de Previsão Climática Contínua para Agricultura de Precisão.
     + Monitora condições meteorológicas em tempo real para ajustar processos agrícolas, como irrigação e proteção de colheitas.
5. **ThingSpeak API**
   * **Tema do Projeto**: Sistema de Monitorização Contínua de Dispositivos IoT (Internet of Things).
     + Coleta e analisa dados de sensores IoT (e.g., temperatura, humidade, localização) continuamente para aplicações como smart cities ou smart homes.
6. **Twitter Streaming API**
   * **Tema do Projeto**: Sistema de Monitoramento Contínuo de Sentimento nas Redes Sociais.
     + Recebe atualizações contínuas sobre tweets relacionados a certos tópicos ou hashtags, permitindo a análise em tempo real de sentimentos ou eventos emergentes.
7. **Alpha Vantage API (dados de mercado financeiro em tempo real)**
   * **Tema do Projeto**: Plataforma de Monitorização Contínua do Mercado de Ações.
     + Recolhe continuamente dados de mercados financeiros e de ações, fornecendo insights em tempo real para decisões de investimento.
8. **AirVisual API**
   * **Tema do Projeto**: Sistema de Monitorização Contínua da Qualidade do Ar.
     + Fornece atualizações contínuas sobre a qualidade do ar em diferentes localizações, útil para projetos de gestão de ambientes urbanos ou interiores.
9. **MQTT (Message Queuing Telemetry Transport)**
   * **Tema do Projeto**: Plataforma de Monitorização Contínua de Sensores Ambientais em Cidades Inteligentes.
     + Utiliza o protocolo MQTT para recolher dados em tempo real de sensores conectados (como sensores de poluição, tráfego, etc.), permitindo a tomada de decisões para gestão urbana.
10. **AWS IoT Core**
    * **Tema do Projeto**: Sistema de Monitorização Contínua de Dispositivos IoT em Ambiente Industrial.
      + Fornece monitorização contínua de sensores industriais para otimização de processos de produção e manutenção preventiva.

Exemplo Concreto de Projeto que Poderias Desenvolver:

**Sistema de Monitoramento de Saúde em Tempo Real**:

* **Conceito**: Uma plataforma de monitoramento de pacientes que coleta dados de sensores biomédicos (e.g., frequência cardíaca, oxigenação do sangue) e analisa em tempo real para identificar eventos críticos.
* **Aquisição de dados**: Sensores biométricos enviam dados para um servidor central.
* **Publicação de dados**: Protocolo de mensagens leves para enviar dados dos sensores para o backend.
* **Armazenamento**: Banco de dados para armazenar os dados dos pacientes e suas métricas ao longo do tempo.
* **Processamento**: Algoritmos detectam situações de alerta e notificam médicos ou familiares.
* **API**: Endpoints RESTful para acessar os dados de saúde, tanto em tempo real quanto históricos.
* **Web Portal**: Interface para médicos visualizarem a condição dos pacientes, com gráficos e alertas de eventos críticos.
* **Aplicativo Móvel**: Opção de receber notificações em tempo real para médicos ou pacientes.

**Projeto: Sistema de Recomendação Musical Personalizado**

**Conceito:**

Uma aplicação que recomenda playlists personalizadas para os utilizadores com base nas suas preferências musicais e no seu histórico de escuta. A aplicação conecta-se à API do Spotify para recolher dados sobre as músicas e artistas favoritos de um utilizador e gera playlists baseadas no humor, género ou estilo que o utilizador preferir em determinado momento.

**Funcionalidades:**

1. **Conexão com a Conta do Spotify**: O utilizador faz login utilizando a conta do Spotify (OAuth 2.0).
2. **Recolha de Dados**: A aplicação obtém informações sobre o histórico de escuta, playlists existentes e artistas/músicas favoritos.
3. **Análise de Preferências**: Baseado nos dados recolhidos, a aplicação identifica padrões, como géneros favoritos, artistas mais ouvidos, momentos do dia em que escuta mais música.
4. **Geração de Playlists**: O utilizador pode selecionar um estado de espírito (e.g., relaxar, treinar, concentrar-se), e a aplicação cria automaticamente uma playlist de acordo com essas preferências e os dados do Spotify.
5. **Partilha de Playlists**: Os utilizadores podem partilhar as playlists geradas com amigos.
6. **Visualização de Estatísticas**: Mostra ao utilizador gráficos sobre as suas preferências musicais, com dados extraídos diretamente do Spotify.

**Ferramentas e Arquitetura:**

* **API**: Spotify Web API (utilizando OAuth 2.0 para autenticação).
* **Backend**: Node.js com Express para a criação da API personalizada.
* **Frontend**: React para a interface web.
* **Banco de Dados**: MongoDB (para armazenar preferências dos utilizadores, playlists geradas, etc.).
* **Processamento**: Utilização de algoritmos de recomendação baseados em machine learning (opcional).
* **Deploy**: Docker para contêineres e Heroku para deployment.
* **Documentação da API**: OpenAPI/Swagger.

**Exemplos de Endpoints:**

* GET /spotify/user-data: Recolhe os dados de perfil e histórico de músicas ouvidas do utilizador.
* POST /spotify/recommend-playlist: Gera uma playlist personalizada.
* GET /spotify/user-stats: Visualização das estatísticas de escuta.

**Iteração 1.1**

|  |  |  |
| --- | --- | --- |
| **ToDo** | **Doing** | **Done** |
| Atribuir as roles a cada membro do grupo e definir o seu trabalho |  | Architect → Ricardo  DevOps → Andre  TeamManager → Violeta  ProdOwner → Rodrigo |
|  | Definir o conceito do nosso projeto | Gestor de Agricultura |
|  | Escolher as ferramentas e arquitetura a usar |  |
| Criar o repositório e o ambiente do trabalho do projeto |  |  |
| Definir as personas e as user Stories (8-12 stories) |  |  |
| Escrever o backlog das conversas / reuniões que tivemos, o que ficou definido em cada uma, quem fez o quê |  |  |
| Criação do changelog, para guardar uma lista dos pull requests e issues de código desenvolvido durante cada uma das iterações |  |  |
| Entrega completa da iteração 1.1 de acordo com as indicações do stor, ex: deve ter a tag 0.1.1 |  |  |

**TO DO’S gerais**

1. **Perceber as roles para definir o membro e atribuir o trabalho:**
   1. **Team Manager** – É aquele que promove a interação e colaboração da equipa, fazendo uma distribuição justa de tarefas a ele mesmo e aos restantes membros do grupo, estando a supervisionar se os membros do grupo estão a ser capazes de realizar o trabalho que lhes foi atribuído e de forma a garantir que o grupo consegue entregar os objetivos de cada iteração dentro do tempo fornecido. É também ele, que será responsabilizado e que terá de defender o grupo na eventualidade de na apresentação da iteração algo não estar a funcionar corretamente, ou no caso de um colega que não consiga realizar a parte do trabalho que lhe foi atribuída este deve tomar a iniciativa e realizar esse trabalho ou então de delegar essa tarefa para outro membro que esteja disponível e com capacidades para tal. Tem por isso uma carga de trabalho contínua ao longo do trabalho, podendo ainda haver picos do esforço que vai ter de oferecer para o projeto por causa das situações acima descritas
   2. **Product owner** – Representa os interesses dos nossos clientes e por isso terá de ter uma compreensão bastante aprofundada do produto que os clientes desejam. Assim sendo estará encarregue de levantar e definir a maioria dos requisitos e funcionalidades iniciais do projeto e em fases posteriores estará encarregue de aceitar ou não incrementos do produto de forma a garantir que estas estão de acordo com os melhores interesses e requisitos levantados pelos stakeholders. Desta forma vai ter um pico de esforço bastante superior aos outros no inicio do projeto e depois esforços situacionais quando tiver de estar a verificar se as soluções entregues cumprem ou não os requisitos para poderem ser adicionadas à main.
   3. **Architect –** Será aquela pessoa que tem uma compreensão mais profunda e detalhada de como funcionamas ferramentas e arquitetura que estão a ser utilizadas no projeto. Sendo assim aquele membro do grupo que será questionado pelo professor o porque de terem sido escolhidas aquelas ferramentas e arquiteturas especificas para desenvolver o projeto, explicando também para os seus colegas de grupo qual será o comportamento esperado destas ferramentas.
   4. **DevOps –** Responsável por garantir que o ambiente de trabalho foi inicializado corretamente, de forma a funcionar em qualquer máquina e não apenas no seu computador pessoal, garantindo também que o subsequente desenvolvimento do código continua estar a funcionar corretamente. Trata-se por isso do membro do grupo que ficará encarregue de analisar o código que está a ser entregue pelos outros membros e resolver eventuais problemas que vão surgindo, e que os outros membros não sejam capazes de os resolver
   5. **Developer –** atributo comum a todos os membros do projeto
2. **Definir aquilo que vai ser o nosso projeto/produto.** Sabemos então que temos de desenvolver uma especificação do produto a qual seja derivada das user Stories e dos requisitos levantados pelo Product Owner
3. **Escolha das ferramentas e arquitetura a usar no desenvolvimento do projeto:**
   1. **Data sources -**
   2. **Backend -**
   3. **Frontend –**
4. **Na criação do repositório git** garantir que segue a estrutura do enunciado, ou seja:
   1. Ter um **README** onde colocamos: 1. os elementos do grupo e quais as roles de cada um; 2.título e breve descrição do projeto e suas capacidades; 3.Link para os diagramas da arquitetura do projeto, bem como uma descrição de como correr e usar a nossa aplicação ; 4.Bookmarks para aceder aos recursos do nosso projeto
   2. Diretório **reports** com o pdf do project specifications
   3. Diretório **minutes**, no qual devemos guardar os minutes do nosso projeto sendo que cada reunião resulta num minute?
   4. Diretório **presentations** com os materiais usados nas apresentações
   5. Diretório com o **source code**
5. Definir quais os objetivos a serem alcançados em cada iteração e o que temos de criar e entregar em cada uma destas para garanti-los.
   1. Cada iteração deve ser entregue de acordo com as indicações dos prof, ex ter a tag correspondente a cada iteração
6. Definir qual o objetivo do sprint da primeira aula sobre o projeto e definir o sprint da próxima aula. Sendo que em cada sprint temos de apresentar pelo menos o seguinte:
   1. Quanto tempo passou desde que nos reunimos sobre o sprint anterior
   2. Ter o backlog criado e atualizado até este último sprint
   3. Bem como um planeamento do que será o backlog para a próxima iteração
   4. E Por fim uma lista dos pull requests e issues de código desenvolvido durante a presente iteração
7. Qual o tema do projeto / que aplicação vamos desenvolver
8. Que API vamos usar, e porque.
9. Quais dados vamos retirar do API, e quais vamos usar

database RT → api → backend → api→ Frontend

Precisamos de dados real time

1/5,25 + 1/3,70 + 1/1,5 = 1,12

= 0,95

cargos

Nao queremos

Ricardo → TeamManager DevOps

Rodrigo → DevOps

Violeta → Product owner

Andre → Product owner

Queremos

Ricardo

Rodrigo Architect, Product Owner

Violeta Architect, DevOps, TeamManager

Andre Architect, DevOps, TeamManager

EUREKAAA

Ideia → Gestor de Agro-Pecuaria

Descrição →  Uma plataforma para monitorização de campos agrícolas com sensores de solo, temperatura e humidade, conectados em tempo real.

Pode sugerir horários ideais para irrigação ou aplicar fertilizantes.

Podemos ajustar a rega caso a meteorologia assim o diga.

Mostra a ‘saúde’ do terreno em tempo real e exibe recomendações.

Permitir que os agricultores manualmente ajustem parâmetros (como nível de irrigação) e enviem dados locais (como observações de pragas).

Minuta #1 - Reuniao 02-10-2024

BrainStorm de ideias

Definição do tema

Descrição do conceito

Definição dos papeis

Ideia simples

→ Ter uma meia de sensores a funfar dados (RT)

sensores camaras temp humidade

Ideia melhor

→

frontend

4 ou 5 graficos com os dados tr

hipotese de ligar ou desligar dispositivos

→