GitHub:
https://github.com/HelioJose10/ProjectFactory

Hélio José – 20190928 e Rúben Passarinho – 20200095

Licenciatura em Engenharia Informática

IADE – Faculdade de Design Tecnologias e Comunicação

Projeto Factory

Professor Fábio Guilherme

31 de maio de 2023

Definição

Ao longo do tempo tivemos um grande aumento no uso de veículos de mobilidade urbana juntamente com uma grande produção variada desses mesmos veículos diante da demanda, da falta de uso de capacetes de segurança pelos usuários de veículos de mobilidade urbana, como bicicletas, patinetes elétricos e motocicletas, em muitas cidades ao redor do mundo. Pode levar a ferimentos graves e até à morte em caso de acidentes de viação, uma vez que a cabeça é uma das partes mais vulneráveis do corpo humano. Muitas vezes, a falta de uso do capacete pode ser atribuída à falta de conscientização ou acesso ao equipamento.

A motivação para o trabalho a ser realizado é a busca de soluções que possam conscientizar sobre a importância do uso do capacete de segurança e torná-lo mais acessível aos usuários de veículos de mobilidade urbana. Estas soluções podem incluir o desenvolvimento de capacetes mais leves e confortáveis e a criação de redes de partilha de capacetes em pontos estratégicos da cidade.

Objetivo

Este projeto visa trazer segurança a estes novos tipos de transporte ecológico, de forma a serem práticos e confortáveis de utilizar no dia-a-dia. Além de práticos, eles terão tecnologia integrada para facilitar a vida do usuário. Sentimos a necessidade de aumentar a segurança no dia a dia trazendo ganhos para a sociedade como:

Permitir que a sociedade escape à inflação dos combustíveis;
Movimente-se com segurança;
Reduzir a taxa de acidentes destes equipamentos;

Ao aumentar a conscientização e o acesso aos capacetes de segurança, espera-se reduzir o número de lesões e mortes relacionadas a acidentes de trânsito envolvendo veículos de mobilidade urbana, tornando as cidades mais seguras para todos os seus usuários.

Solução

O capacete terá a possibilidade de detetar a velocidade que o utilizador está, bem como a velocidade média, detetar eventuais quedas, a sua localização, entre outros... Tudo isto com a ajuda de uma aplicação Android de forma a poder salvaguardar e consultar a informação do capacete, bem como ver os seus percursos percorridos.

Funcionalidades não implementadas

No início do projeto o nosso grupo, projetou algumas funcionalidades para o capacete que não foram alcançadas por grande influência de peças fundamentais que não tivemos acesso, como havia sido programado.

- Módulo Grove - Acelerômetro Digital de 6 eixos e giroscópio (LSM6DS3) - SeeedAcelerômetro/Giroscópio.

(que usaríamos para controlar a velocidade do veículo e detetar eventuais quedas do capacete);

- ICQUANZX GY-NEO6MV2 NEO- 6M GPS - módulo de controlo de voo (3 V-5 V, antena de cerâmica superforte para Arduíno EEPROM APM 2.5).

(usaríamos para detetar e guardar a geolocalização do utilizador do capacete);

- WOWOWO 1S Módulo indicador de ecrã azul com capacidade de bateria de lítio de nível de potência de 3,7 V.

(usaríamos para indicar a bateria do capacete);

- Polymer Lithium Ion Battery - 3.7v 850mAh.

(bateria do capacete);

- Placa de ensaio ''Bread board'' 400 contactos.

(Placa principal onde tudo se iria interligar);

Por não ter acesso aos componentes citados anteriormente não foi possível desenvolver o android por completo a que toca:

- Mapa: percursos de um determinado destino escolhido pelo utilizador e o tempo estimado bem como a distancia em km;

- TechHelmet: botão para se conectar ao capacete, ativar o modo noturno do capacete, obter informações da bateria, obter percursos já realizados e por realizar, ativar modo SOS, ativar alerta de velocidade máxima;

- Percursos: Obter dados específicos sobre percursos já realizados, sugestão de novos percursos, ETC;

Implementação

Descrição de APIs em PHP

Cada API executa uma função específica e interage com um banco de dados MySQL para executar operações relacionadas a dados. As APIs são detalhadas abaixo.

- **API 1: Verificação de credenciais de usuário**

Esta API recebe os parâmetros "email" e "password" via pedido GET e verifica se estão presentes. Se algum dos parâmetros estiver faltando, a API retornará uma resposta de erro com o código de status HTTP 400 (Solicitação incorreta) e uma mensagem no formato JSON indicando que o e-mail ou senha está faltando.

A API então estabelece uma conexão com o banco de dados MySQL usando as informações de host, banco de dados, usuário, senha e charset fornecidas. Se a conexão falhar, a API retornará uma resposta de erro com o código de status HTTP 500 (Erro Interno do Servidor) e uma mensagem no formato JSON indicando a falha de conexão.

A API prepara uma consulta SQL para selecionar o ID do usuário com base no e-mail e na senha fornecidos. Os valores de e-mail e senha são vinculados à consulta preparada usando a função "bind_param" e a consulta é executada.

O resultado da consulta é obtido usando o método "fetch_assoc" e armazenado na variável $user. Se o usuário existir, a API retornará uma resposta no formato JSON com o ID do usuário. Caso contrário, ele retorna uma resposta de erro com o código de status HTTP 404 (Não encontrado) e uma mensagem no formato JSON indicando que o e-mail ou senha é inválido.

- **API 2: Consulta à distância**

Essa API recebe o parâmetro "id" via solicitação GET e verifica se ele está presente. Se estiver faltando, a API retornará uma resposta de erro com o código de status HTTP 400 (Solicitação incorreta) e uma mensagem no formato JSON indicando que o parâmetro está faltando.

A API então estabelece uma conexão com o banco de dados MySQL usando as informações de conexão fornecidas.

A API prepara uma consulta SQL para selecionar os valores de distância, data e hora da tabela "distância" para um determinado ID de usuário. A consulta é executada e o resultado é armazenado no objeto $result.

Se houver registros retornados pela consulta, a API itera sobre os resultados, armazena cada linha em uma matriz associativa chamada $data e, finalmente, retorna a matriz $data no formato JSON.

Se nenhum registro for retornado, a API retornará uma resposta de erro com o código de status HTTP 404 (Não encontrado) e uma mensagem no formato JSON indicando que nenhum dado correspondente ao ID de usuário fornecido foi encontrado.

- **API 3: Inserindo valores de distância**

Essa API aceita solicitações GET e POST e espera os parâmetros "value" e "id". Ele verifica se os parâmetros estão presentes obtendo seus valores do método POST e do método GET.

A API então estabelece uma conexão com o banco de dados MySQL usando as informações de conexão fornecidas.

A API verifica se o ID de usuário fornecido existe no banco de dados antes de prosseguir. Se o ID não existir, a API retornará uma resposta informando que o ID do usuário não existe e que o valor não foi inserido.

Se o ID existir, a API obtém a data e a hora atuais e prepara uma consulta SQL para inserir o valor da distância, o ID do usuário e a data/hora na tabela "distância". A consulta é executada e, se bem-sucedida, a API retorna uma resposta indicando que o valor foi inserido com êxito na tabela. Caso contrário, ele retorna uma resposta de erro com o código de status HTTP 500 (Erro interno do servidor) e uma mensagem indicando que ocorreu um erro ao inserir o valor na tabela.

- **API 4: Registro de novo usuário**

Esta API recebe os parâmetros "nome", "e-mail", "celular" e "senha" via solicitação GET e verifica se todos os parâmetros estão presentes. Se algum parâmetro estiver faltando, a API retornará uma resposta de erro com o código de status HTTP 400 (Solicitação incorreta) e uma mensagem no formato JSON indicando que os campos obrigatórios estão faltando.

A API então estabelece uma conexão com o banco de dados MySQL usando as informações de conexão fornecidas.

A API prepara uma consulta SQL para inserir os valores de nome, e-mail, telefone e senha na tabela "usuários". Os valores são vinculados à consulta preparada usando a função "bind_param" e a consulta é executada.

Se a consulta afetar uma ou mais linhas (ou seja, se o usuário for registrado com êxito), a API retornará uma resposta no formato JSON contendo a ID do usuário recém-registrado. Caso contrário, a API retorna uma resposta de erro com o código de status HTTP 500 (Erro interno do servidor) e uma mensagem no formato JSON indicando que ocorreu um erro ao registrar o usuário.

- **API 5: Armazenando dados POST em um arquivo**

Esta API aceita pedidos POST e armazena os dados recebidos no ficheiro "ip.txt". A API abre o arquivo no modo de gravação, exclui o conteúdo existente, grava os dados POST no arquivo e fecha o arquivo.

Nenhuma verificação de erro ou validação de dados é executada nesta API. O principal objetivo é armazenar os dados recebidos do POST em um arquivo.

As APIs descritas acima fornecem diferentes funcionalidades, como verificar as credenciais do usuário, consultar e inserir dados no banco de dados MySQL, registrar novos usuários e armazenar dados em um arquivo. Cada API implementa uma lógica específica e interage com o banco de dados para executar as operações desejadas. É importante notar que, para o bom funcionamento das APIs, é necessário fornecer as informações de conexão corretas, como host, banco de dados, usuário, senha e charset.

Descrição do Arduino ESP32

O código fornecido destina-se ao Arduino ESP32, que é uma placa de desenvolvimento baseada no microcontrolador ESP32. O ESP32 é amplamente utilizado em projetos de IoT (Internet of Things) devido à sua capacidade de se conectar a redes Wi-Fi e Bluetooth, bem como sua versatilidade e recursos avançados. Este relatório irá analisar e explicar o código fornecido para o Arduino ESP32.

- **Verificando o parâmetro "id":** O código verifica se o parâmetro "id" está presente na solicitação GET. Se não estiver presente, retorna uma resposta de erro com o código de status HTTP 400 (Solicitação incorreta) e uma mensagem no formato JSON indicando que o parâmetro está ausente. Essa verificação é importante para garantir que o código funcione corretamente e evite erros de execução.

- **Conexão com o banco de dados:** O código estabelece uma conexão com o banco de dados MySQL. Ele usa as informações de host, banco de dados, usuário, senha e charset fornecidas para fazer a conexão. Se ocorrer algum erro

durante a conexão, uma resposta de erro será retornada com o código de status HTTP 500 (Erro Interno do Servidor) e uma mensagem no formato JSON indicando que a conexão falhou.

- **Consulta ao banco de dados:** depois que a conexão é estabelecida com êxito, o código prepara uma consulta SQL para obter informações do banco de dados. A consulta é baseada no parâmetro "id" fornecido na solicitação GET. O código usa a função "bind_param" para vincular o valor do parâmetro à consulta preparada. A consulta é então executada usando o método "execute".

- **Processamento de resultados de consulta:** o código verifica se a consulta retornou algum resultado. Se os registros forem encontrados, os dados serão buscados usando o método "fetch_assoc" e armazenados em uma matriz associativa chamada "data". Os dados são então convertidos para o formato JSON usando a função "json_encode" com a opção JSON_UNESCAPED_UNICODE para preservar os caracteres Unicode. O JSON resultante é retornado como resposta.

- **Tratamento de erros:** Se a consulta não retornar nenhum resultado, ou seja, nenhum registro encontrado, uma resposta de erro será retornada com o código de status HTTP 404 (Não encontrado) e uma mensagem no formato JSON indicando que nenhum dado correspondente ao parâmetro "id" foi encontrado.

- **Fechando a conexão:** Após o processamento, o código fecha a conexão com o banco de dados e termina a execução.

O código fornecido para o Arduino ESP32 demonstra como realizar consultas a um banco de dados MySQL a partir do microcontrolador. Ele usa a biblioteca MySQLi para estabelecer a conexão e executar a consulta. O código é útil para recuperar informações específicas do banco de dados com base no parâmetro "id" fornecido na solicitação GET. É importante garantir que as informações de conexão do banco de dados estejam corretas e que os erros sejam tratados adequadamente para garantir que o código funcione corretamente.

Android

Uma implementação de um servidor web Java leve chamado TinyWebServer. O servidor é capaz de lidar com solicitações HTTP e fornecer respostas apropriadas.

- **Estrutura do projeto:**

- TinyWebServer.java: Principal implementação do servidor web.

- MainActivity.java: Classe que representa a atividade principal do aplicativo Android.
- login.java: Classe que representa a atividade de login do aplicativo Android.
- register.java: Classe que representa a atividade de registro do aplicativo Android.

- **Principais características:**

- Inicialização do servidor: O servidor é iniciado criando uma instância da classe ServerSocket e definindo um endereço IP e uma porta para escutar as solicitações recebidas.
- Processamento de solicitações: O servidor aceita solicitações HTTP de clientes, lê os dados da solicitação, analisa os cabeçalhos e parâmetros e determina a ação apropriada a ser tomada com base na solicitação recebida.
- Roteamento de solicitações: O servidor encaminha as solicitações para o local correto com base na URL recebida. Se o URL corresponder à raiz do servidor, o servidor devolve o ficheiro "index.html". Caso contrário, o servidor procura o ficheiro correspondente ao URL e devolve o seu conteúdo.
- Suporte a arquivos estáticos: O servidor é capaz de retornar arquivos estáticos, como HTML, imagens (JPEG, PNG) e vídeos (MP4), com base na solicitação do cliente.
- Suporte para solicitações POST: O servidor lida com solicitações POST enviadas por clientes e processa os dados recebidos.
- Envio de dados para uma URL externa: O servidor é capaz de enviar dados recebidos via POST para uma URL externa especificada.
- Tela de login: A atividade "login" permite que o usuário insira um e-mail e senha para fazer login.
- Chamada de API de login: A classe "LoginTask" executa uma chamada de API para verificar as credenciais do usuário. A resposta da API é analisada para determinar se o login foi bem-sucedido.
- Redirecionamento após o login: Se o login for bem-sucedido, o aplicativo redireciona o usuário para a atividade principal ("MainActivity").
- Tela de cadastro: A atividade "cadastrar" permite que o usuário insira seus dados cadastrais, como nome, e-mail, telefone e senha, para criar uma conta.
- Chamada de API de registro: A classe "register" executa uma chamada de API para enviar os dados de registro para o servidor. A resposta da API é manipulada para exibir uma mensagem de êxito ou erro na interface do usuário.

- **Requisitos cumpridos:**

- Aceitar solicitações HTTP recebidas.
- Ler e analisar os cabeçalhos e parâmetros do pedido.
- Roteamento adequado de solicitações com base na URL recebida.
- Suporte para arquivos estáticos, como HTML, imagens e vídeos.
- Tratamento dos pedidos POST e tratamento dos dados recebidos.
- Envio de dados para um URL externo especificado.

- Implementação de uma tela de login para autenticação do usuário.
- Ligue para a API de login para verificar as credenciais do usuário.
- Redirecionar para a atividade principal após o login bem-sucedido.
- Implementação de uma tela de registro para criação de conta de usuário.
- Ligue para a API de registro para enviar os dados de registro para o servidor.
- Manipulação da resposta da API de registro para exibir uma mensagem de sucesso ou erro.

Além desses requisitos, o código também inclui outras funcionalidades, como:

- Verificação e solicitação de permissões de escrita e leitura em armazenamento externo.
- Criação de um arquivo "index.html" com informações de IP do dispositivo.
- Inicialização de um servidor web baseado no IP e porta fornecida.
- Uso da biblioteca Volley para fazer solicitações POST para uma URL externa.
- Tratamento de respostas de solicitação POST e exibição de mensagens de sucesso ou erro.

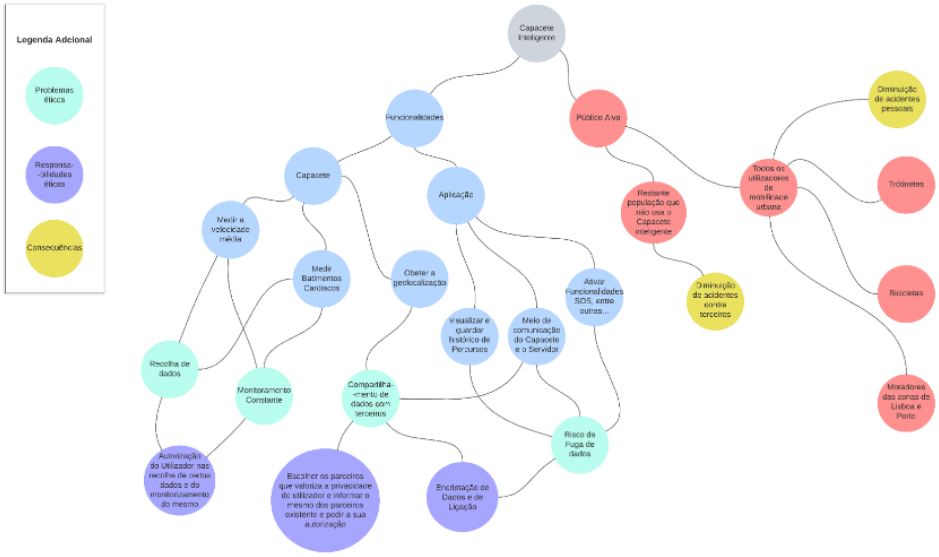
- **Considerações finais**

O código fornecido combina a implementação do TinyWebServer com um aplicativo Android que executa o login do usuário, o registro da conta e as interações do servidor. Ele demonstra a integração entre um servidor Web personalizado e um aplicativo cliente, fornecendo funcionalidades como autenticação, processamento de solicitações HTTP e manipulação de arquivos estáticos.

É importante observar que a implementação fornecida pode ter dependências externas, como bibliotecas de terceiros ou arquivos de layout específicos do projeto Android, que não estão incluídas no código fornecido.

Para compilar e executar esse código, você precisa configurar um ambiente de desenvolvimento Android e configurar as dependências apropriadas.

Ética e Deontologia Profissional



Introdução

Neste relatório pretende-se avaliar possíveis questões éticas relativas à privacidade e segurança, bem como as responsabilidades inerentes à prevenção de fugas de dados, à partilha de informação com terceiros e à garantia da segurança dos utilizadores. A análise basear-se-á na Ética das Virtudes, na perspetiva do Consequencialismo de John Stuart Mill e na Deontologia de Immanuel Kant.

Ética da Virtude

A Ética da Virtude sugere que a prática moral e ética provém do desenvolvimento de um caráter virtuoso. Assim, devemos priorizar a honestidade, integridade e responsabilidade ao lidar com os dados dos usuários. O monitoramento constante e a coleta de dados são práticas que podem ser questionáveis do ponto de vista da Ética da Virtude, pois podem comprometer a confiança e o respeito mútuo. Além disso, devemos ter em conta a virtude da prudência, de forma a evitar a partilha de dados com terceiros e mitigar os riscos de roubo de dados.

Consequencialismo - John Stuart Mill

O conseqüencialismo, defendido por Mill, avalia as ações morais com base em seus resultados. A ação correta é aquela que maximiza a felicidade geral. Por conseguinte, a segurança dos utilizadores e a proteção dos seus dados não devem ser comprometidas por práticas que possam levar a consequências negativas, como o roubo de dados ou acidentes causados pela utilização de aplicações durante as viagens. Temos de garantir que as suas práticas geram mais benefícios do que danos para os utilizadores.

Deontologia - Immanuel Kant

A Deontologia Kantiana sugere que o que torna uma ação moral ou não é o princípio que a orienta, independentemente do resultado. Nesta perspetiva, o respeito pela privacidade e pelos dados dos utilizadores é um dever prioritário para nós. Tal inclui não partilhar dados com terceiros e evitar práticas que possam conduzir à fuga de dados. Além disso, é nossa obrigação mantê-lo seguro, independentemente das consequências para a própria empresa.

Conclusão

Em conclusão, com base nas três perspetivas éticas, é fundamental adotar práticas que respeitem a privacidade e a segurança dos utilizadores, evitando a recolha desnecessária de dados, a monitorização excessiva e a partilha de dados com terceiros. Além disso, temos de assumir a responsabilidade de prevenir a fuga de dados e garantir a segurança dos utilizadores, respeitando assim os princípios da integridade, prudência e responsabilidade.