TRABALHO

Introdução

O uso da rastreabilidade iniciou-se da necessidade de identificar individualmente cada item/objeto, garantindo uma gestão mais eficiente,

Foi investigado a necessidade de realizar uma rastreabilidade que seja eficiente e segura, na qual hoje é um problema a realização da rastreabilidade de objetos serem feitas de forma manual, ocasionando uma gestão ineficientes e consequentemente prejuízos financeiros, como, roubo, extravio e também gastos desnecessários com vistorias e aferição de todo patrimônio.
Diante desses problemas foi proposta a utilização de tecnologia para realizar a identificação de modo eletrônico, automatizando o processo de identificação e possibilitando agilidade e praticidade para realização do controle de patrimônio.
Visando desenvolver uma solução com custos mais acessíveis foi feito uma pesquisa elencando tecnologias que possibilitariam o controle e gestão de patrimônio. Para isso um sistema será desenvolvido e implantado para monitorar e verificar a sua eficiência, de forma a aumentar a segurança, evitando extravios e perca de patrimônio.

Desenvolver um sistema de rastreabilidade para controle de patrimônio utilizando a tecnologia RFID de baixo custo, possibilitando a rastreabilidade e identificação dos patrimônios nos respectivos ambientes.

Desenvolver uma aplicação de monitoramento em tempo real do patrimônio, melhorar a gestão de patrimônio

Desenvolver um sistema de identificação, possibilitando melhoramento na segurança, minimizando extravio de patrimônio;

Identificar características e local do referido patrimônio.

Possibilidades Rastreabilidade de Patrimônio

1. Rastrear sala individual
2. Rastrear Unidade Operacional (filial)
3. Aumentar segurança contra roubo ou extravio de equipamento eletrônicos, fazendo validação cruzada, em caso de movimentação de objeto faz necessário o colaborador fazer leitura do cartão pessoal. Assim associando a movimentação de patrimônio e colaborador, em caso de não identificação do colaborador, o sistema emite um alerta como possível extravio
4. Detecção automática ao movimentar objetos de ambientes
5. Detecção através da realização da leitura manual
6. Categorizar salas com equipamento de alto valor agregado

TECNOLOGIA

• Cartão RFID (crachá individual), ou qualquer outro tipo de tag (para realizar leitura) considerando movimentações por funcionários.

• Tag de identificação automática (objeto passar pela porta automaticamente realiza leitura(entrada/saída) do ambiente.

• Rastrear e relacionar Usuário e ambiente

Utilizar struct e manipulação em arquivo

OBS: você terá controle e rastreabilidade de patrimônio bem como a gestão de usuários.

Considerar temática de uma escola/faculdade

Definir as Estruturas (Structs):

```
// Estrutura para representar um patrimônio
struct Patrimonio {
    int id;
    char descricao[100];
    char localizacao[50];
};

// Estrutura para representar um usuário
struct Usuario {
```

```
    int id;
    char nome[50];
    char cargo[50];
};
```

Implementar Funções de Manipulação de Arquivo:

```
// Função para salvar um patrimônio em um arquivo
void salvarPatrimonio(struct Patrimonio patrimonio) {
    FILE *arquivo;
    arquivo = fopen("patrimonios.txt", "a");

    if (arquivo != NULL) {
        fprintf(arquivo, "%d;%s;%s\n", patrimonio.id, patrimonio.descricao, patrimonio.localizacao);
        fclose(arquivo);
    }
}

// Função para salvar um usuário em um arquivo
void salvarUsuario(struct Usuario usuario) {
    FILE *arquivo;
    arquivo = fopen("usuarios.txt", "a");

    if (arquivo != NULL) {
        fprintf(arquivo, "%d;%s;%s\n", usuario.id, usuario.nome, usuario.cargo);
        fclose(arquivo);
    }
}
```

Desenvolver Funcionalidades Principais:

```
// Função para adicionar um novo patrimônio
void adicionarPatrimonio() {
    struct Patrimonio novoPatrimonio;
    printf("Informe o ID do patrimônio: ");
    scanf("%d", &novoPatrimonio.id);
    printf("Informe a descrição do patrimônio: ");
    scanf("%s", novoPatrimonio.descricao);
    printf("Informe a localização do patrimônio: ");
    scanf("%s", novoPatrimonio.localizacao);
    salvarPatrimonio(novoPatrimonio);
    printf("Patrimônio adicionado com sucesso!\n");
}

// Função para adicionar um novo usuário
void adicionarUsuario() {
    struct Usuario novoUsuario;
    printf("Informe o ID do usuário: ");
    scanf("%d", &novoUsuario.id);
    printf("Informe o nome do usuário: ");
```

```
    scanf("%s", novoUsuario.nome);
    printf("Informe o cargo do usuário: ");
    scanf("%s", novoUsuario.cargo);
    salvarUsuario(novoUsuario);
    printf("Usuário adicionado com sucesso!\n");
}
```

Menu de Interface de Usuário:

```
int main() {
    int opcao;
    while (1) {
        printf("Selecione uma opção:\n");
        printf("1. Adicionar patrimônio\n");
        printf("2. Adicionar usuário\n");
        printf("3. Sair\n");
        printf("Opção: ");
        scanf("%d", &opcao);

        switch (opcao) {
            case 1:
                adicionarPatrimonio();
                break;
            case 2:
                adicionarUsuario();
                break;
            case 3:
                return 0;
            default:
                printf("Opção inválida.\n");
        }
    }
    return 0;
}
```

Busca de Patrimônios e Usuários:

```
// Função para buscar um patrimônio por ID
void buscarPatrimonioPorID(int id) {
    FILE *arquivo;
    arquivo = fopen("patrimonios.txt", "r");

    if (arquivo != NULL) {
        struct Patrimonio patrimonio;
        while (fscanf(arquivo, "%d;%[^;];%s\n", &patrimonio.id, patrimonio.descricao,
patrimonio.localizacao) != EOF) {
            if (patrimonio.id == id) {
                printf("ID: %d, Descrição: %s, Localização: %s\n", patrimonio.id, patrimonio.descricao,
patrimonio.localizacao);
                fclose(arquivo);
```

```
            return;
        }
    }
    printf("Patrimônio não encontrado.\n");
    fclose(arquivo);
  }
}

// Função para buscar um usuário por ID
void buscarUsuarioPorID(int id) {
  FILE *arquivo;
  arquivo = fopen("usuarios.txt", "r");

  if (arquivo != NULL) {
    struct Usuario usuario;
    while (fscanf(arquivo, "%d;%[^;];%s\n", &usuario.id, usuario.nome, usuario.cargo) != EOF) {
      if (usuario.id == id) {
        printf("ID: %d, Nome: %s, Cargo: %s\n", usuario.id, usuario.nome, usuario.cargo);
        fclose(arquivo);
        return;
      }
    }
    printf("Usuário não encontrado.\n");
    fclose(arquivo);
  }
}
```

Edição de Dados de Patrimônios e Usuários:

```
// Função para editar informações de um patrimônio por ID
void editarPatrimonio(int id) {
  // Implemente a lógica para editar as informações de um patrimônio
}

// Função para editar informações de um usuário por ID
void editarUsuario(int id) {
  // Implemente a lógica para editar as informações de um usuário
}
```

Exclusão de Patrimônios e Usuários:

```
// Função para excluir um patrimônio por ID
void excluirPatrimonio(int id) {
  // Implemente a lógica para excluir um patrimônio
}

// Função para excluir um usuário por ID
void excluirUsuario(int id) {
  // Implemente a lógica para excluir um usuário
}
```

```
printf("4. Buscar patrimônio por ID\n");
printf("5. Buscar usuário por ID\n");
printf("6. Editar patrimônio por ID\n");
printf("7. Editar usuário por ID\n");
printf("8. Excluir patrimônio por ID\n");
printf("9. Excluir usuário por ID\n");
```

Validações na Adição de Patrimônios e Usuários:

```
// Exemplo de validação para campos vazios ou com espaços em branco
if (strlen(novoPatrimonio.descricao) == 0 || strlen(novoPatrimonio.localizacao) == 0) {
   printf("Erro: Descrição e localização não podem estar em branco.\n");
   return;
}
```

Validações na Busca por ID:

```
// Exemplo de validação para verificar se um patrimônio com o ID informado existe
if (!patrimonioComIDExiste(id)) {
   printf("Patrimônio com o ID %d não encontrado.\n", id);
   return;
}
```

Validações na Edição de Dados:

```
// Exemplo de validação para campos vazios ou com espaços em branco na edição
if (strlen(novoPatrimonio.descricao) == 0 || strlen(novoPatrimonio.localizacao) == 0) {
   printf("Erro: Descrição e localização não podem estar em branco.\n");
   return;
}
```

Validações na Exclusão de Dados:

```
// Exemplo de confirmação de exclusão
printf("Tem certeza de que deseja excluir o patrimônio com o ID %d? (S/N): ", id);
char resposta;
scanf(" %c", &resposta);
if (resposta != 'S' && resposta != 's') {
   printf("Exclusão cancelada.\n");
   return;
}
```