INVENTÁRIO AUTOMATIZADO COM RFID.

Aluno(a): Rafael Antônio Losi

Orientador: Mauro Marcelo Mattos

FURB
UNIVERSIDADE DE BLUMENAU

Roteiro

- Objetivos
- Fundamentação teórica;
- Tecnologia utilizada;
- Trabalhos Correlatos;
- Principais Requisitos;
- Desenvolvimento e arquitetura;
- Demonstração do software.
- Resultados e Conclusões;

Introdução

“A maior dificuldade relatada na gestão patrimonial é a mudança de localização dos bens nos órgãos”

(LEÃO; SOUSA, 2014, p. 13)

Objetivo geral

Desenvolvimento de uma solução para controle patrimonial e inventário utilizando a tecnologia de RFID.

Objetivos Específicos

a) desenvolver um leitor de etiquetas RFID com arduino e modulo de rádio frequência;
b) desenvolver um sistema web para gestão patrimonial;
c) criar um cenário de testes para validar a solução.

Fundamentação Teórica

“O controle patrimonial [...] compreende uma sequência de atividades que iniciam com a aquisição e terminam com a retirada ou alienação do bem do patrimônio”

“toda organização deve possuir em sua estrutura administrativa uma unidade ou departamento responsável pelo controle do patrimônio”.

(FIJOR, 2014)

SIG na gestão patrimonial

“a essência do planejamento e do controle é a tomada de decisões, que depende de informações oportunas e confiáveis”.

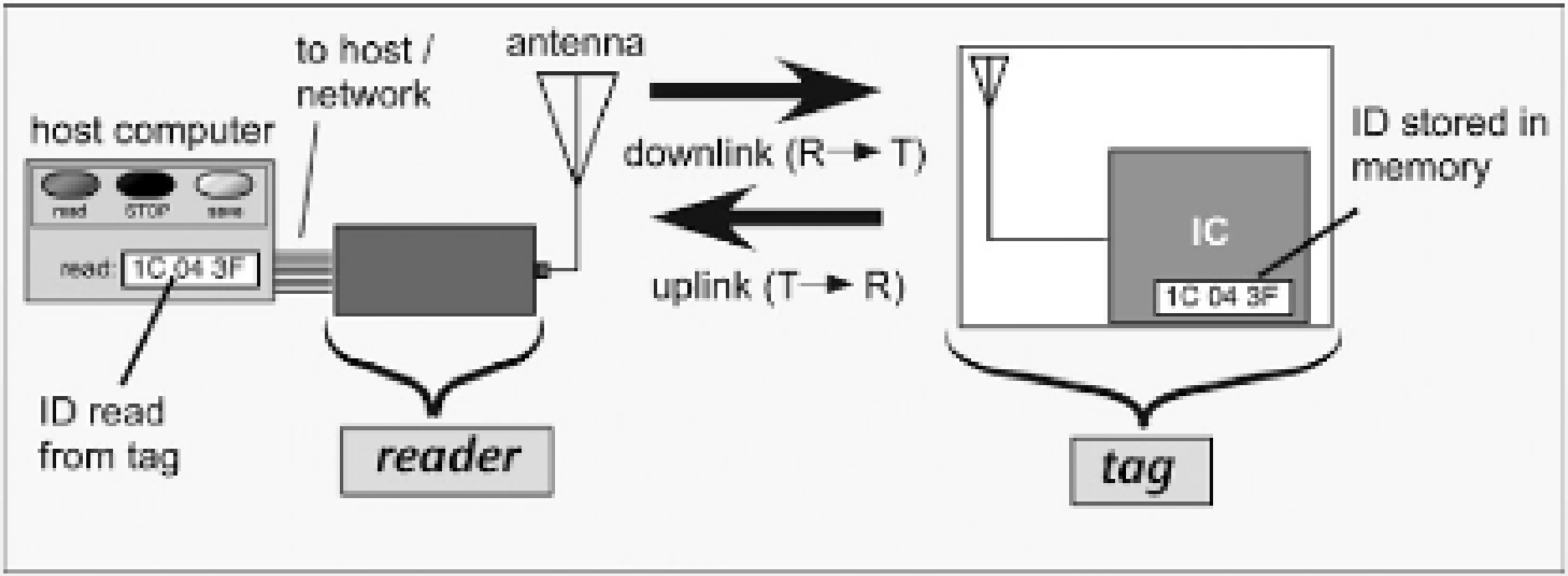
(PORTO; BANDEIRA, 1999)

SIG na gestão patrimonial

“a competitividade de uma empresa é determinada pela qualidade de seus recursos, pelo conhecimento que é capaz de produzir e pela capacidade de aplicar a ciência, a tecnologia e o conhecimento na produção de bens e serviços cada vez mais eficientes”

(DALFOVO, 2004).

Identificação por radiofrequência (RFID)



Fonte: Dobkin (2013).

Identificação por radiofrequência (RFID)

Dados técnicos não bastam para decidir com qual frequência trabalhar, é necessário avaliar o ambiente no qual será utilizado para evitar interferências e dificuldades de leitura o que atrapalharia o funcionamento da solução.

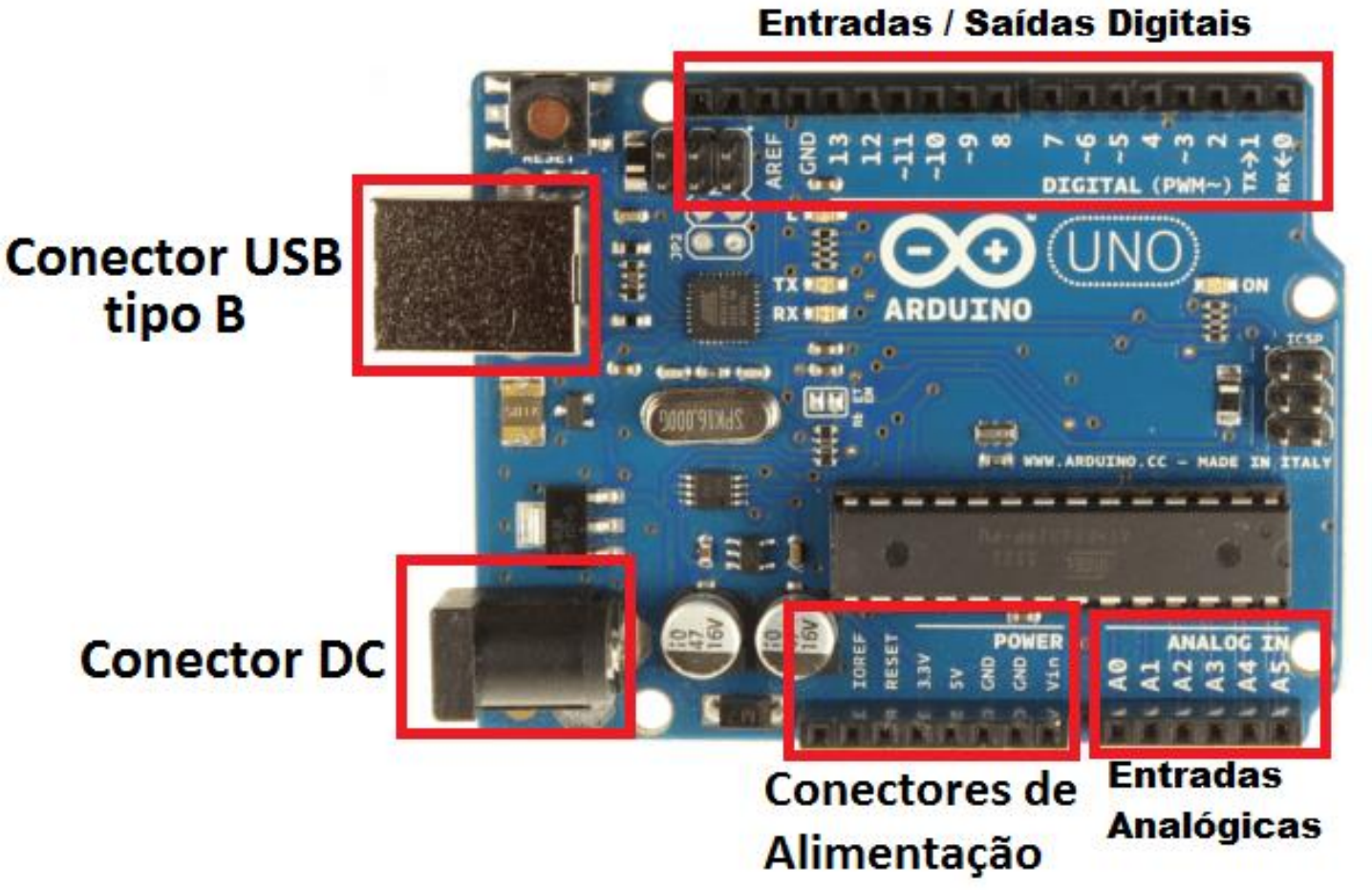
(DIAS; BADALEI, 2012)

Frequências de operação

Banda de Frequência	Características	Aplicações Típicas
Baixa: 100 a 500 Khz	- Faixa de curta até média leitura - Baixo custo - Baixa velocidade de leitura	- Controle de acesso - Identificação de animais - Controle de inventário
Média: 10 a 15 MHz	- Faixa de curta até média leitura - Potencialmente de baixo custo - Média velocidade de leitura	- Controle de acesso - Smart Card
Alta: 850 a 950 MHz e 2,4 a 5,8 GHz	- Faixa de larga leitura - Alto custo - Alta velocidade de leitura - Linha de visão requerida	- Monitoração de veículos em estradas

Fonte: Narciso (2008).

ARDUINO


Entradas / Saídas Digitais
Conector USB tipo B
Conector DC
Conectores de Alimentação
Entradas Analógicas
DIGITAL (PWM~)
POWER
ANALOG IN
UNO
ARDUINO
WWW.ARDUINO.CC – MADE IN ITALY

Módulo RC522

Trabalhos Correlatos

a) Protótipo de um sistema de identificação eletrônica de animais através de rádio frequência (SANTOS, 2001).
b) Sistema de informação aplicado à cadeia de suprimentos utilizando tecnologia RFID (ZEINDIN, 2005).
c) protótipo de um sistema para monitoração de carros de competições, utilizando rádio frequência (DIENER, 2002).

O trabalho de Santos (2001)

Aplicar a tecnologia RFID para aquisição de dados em atividades produtivas na área de produção animal.

Manter um histórico do animal desde o nascimento até o abate.

Leitor Utilizado: SCL05 da Olimex.

Frequência de operação: 128 kHz e 9600 bps

O trabalho de Diener (2002)

Software que monitore carros de corrida.

Benefícios:

a) aumento da produtividade;
b) redução de erros;
c) melhoria nas condições de segurança.

Módulo: PICLAB 4A. (Silva Júnior).

Frequência de operação: 315 Mhz.

O trabalho de Zeindin (2005)

Controlar a quantidade de produtos em estoque em toda cadeia produtiva.

Leitor: Z3070 (ZEBEK).

Frequência de operação: 13.56 MHZ.

Trabalhos correlatos

	Santos (2001)	Diener (2002)	Zeindin (2005)
Ambiente de desenvolvimento	Delphi 5	PicBasicPro	Visual Studio 2003
Linguagem	Delphi	Basic	C#.Net
Aplicação	Rastreamento Animal	Corrida de Carros	Cadeia de Suprimentos
Frequência de Operação	128 khz	315 Mhz	13.56 Mhz
Distância de Operação	50 mm	50 m	100 mm

Requisitos Funcionais

RF02: O sistema deverá exibir ao usuário uma lista de bens sob sua responsabilidade.

RF04: O sistema deverá permitir que o usuário transfira seus bens a outro local ou departamento.

RF05: O sistema deverá registrar um pedido de aprovação da transferência caso o usuário responsável pelo destino da transferência não seja o mesmo.

Requisitos Funcionais

RF06: O sistema deve permitir o usuário consultar as transferências aguardando sua aprovação

RF07: O sistema deve permitir o usuário realizar a conferência dos bens em período de inventário

RF15: O sistema deve permitir ao administrador identificar um bem através de uma *tag* RFID.

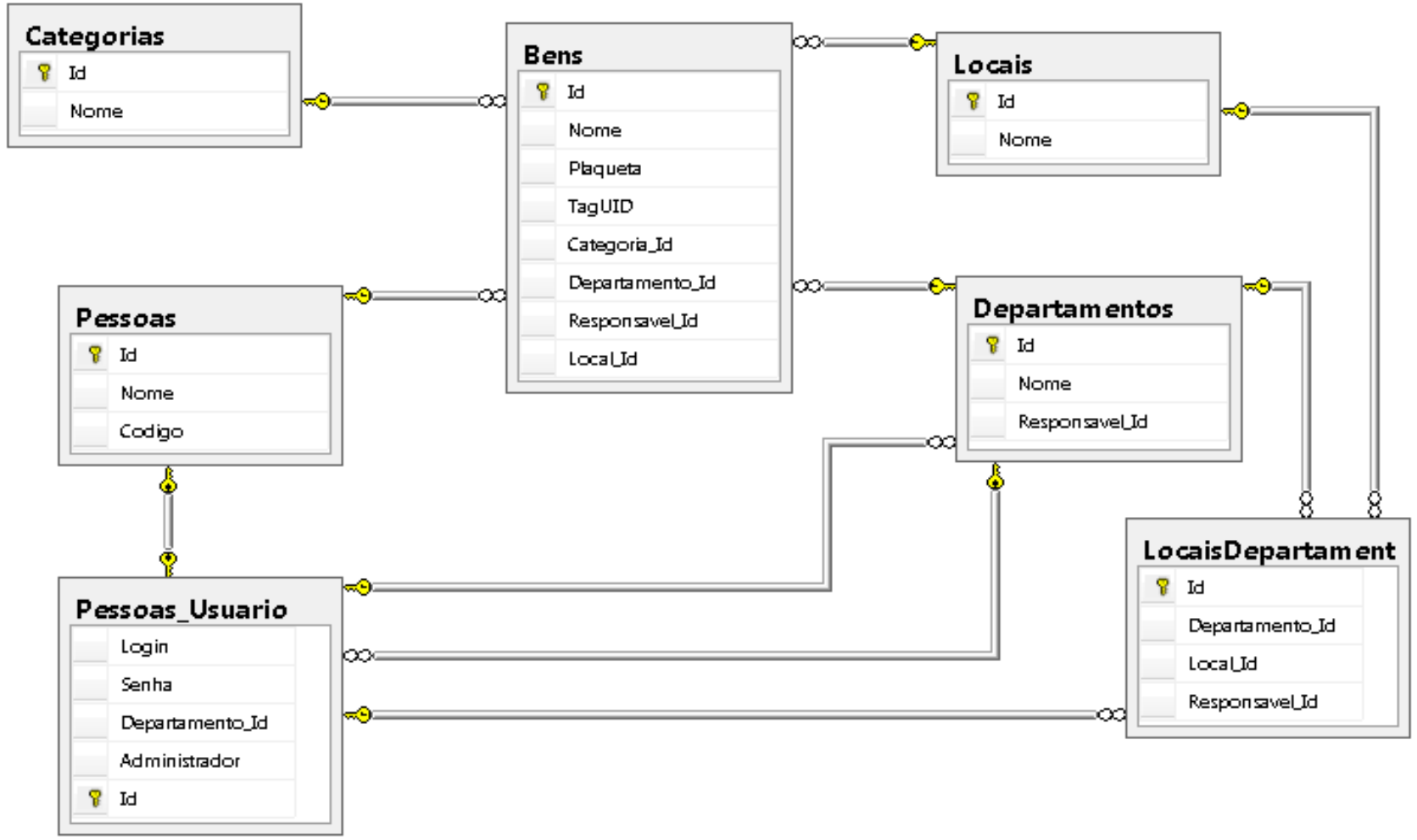
Requisitos não funcionais

RNF01: O sistema deverá ser desenvolvido em com a linguagem C# e framework .Net.

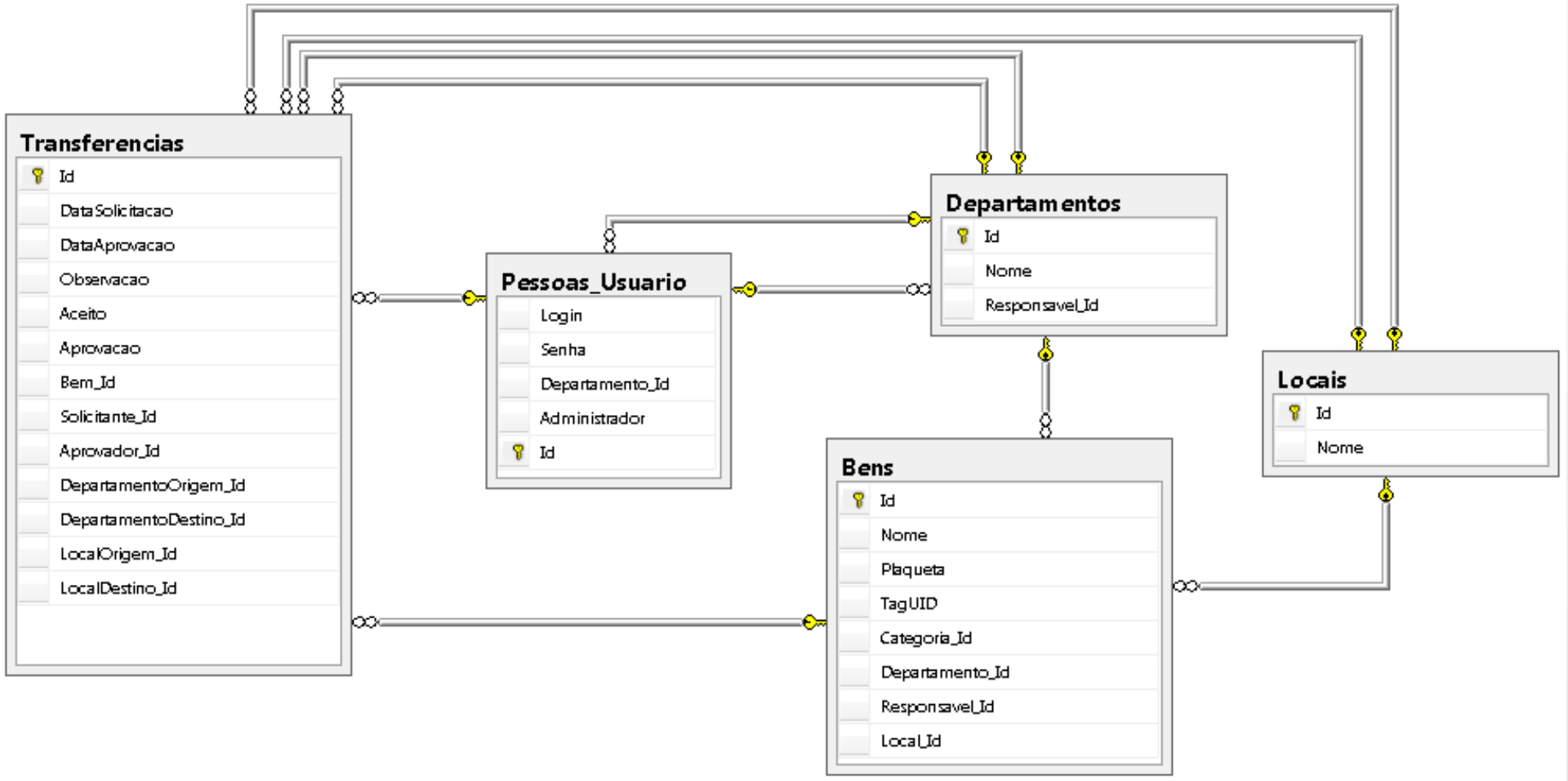
RNF02: O sistema deverá utilizar o banco de dados SQL Server 2014 Express.

RNF05: O leitor de tags RFID deve ser desenvolvido com placa Arduino UNO e modulo de radiofrequência RC522.

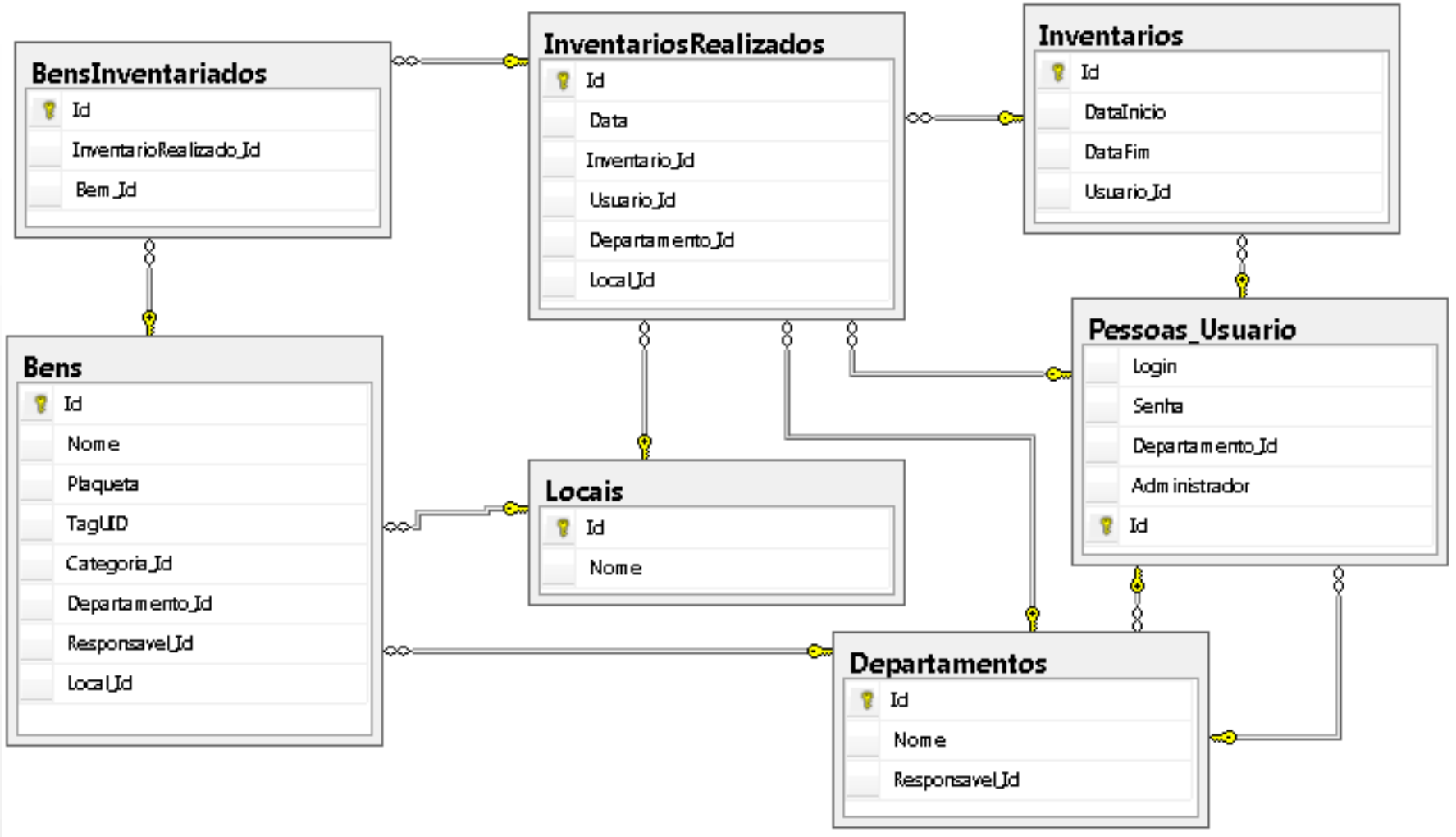
MER dos cadastros



MER entidades transferência



MER entidades inventário



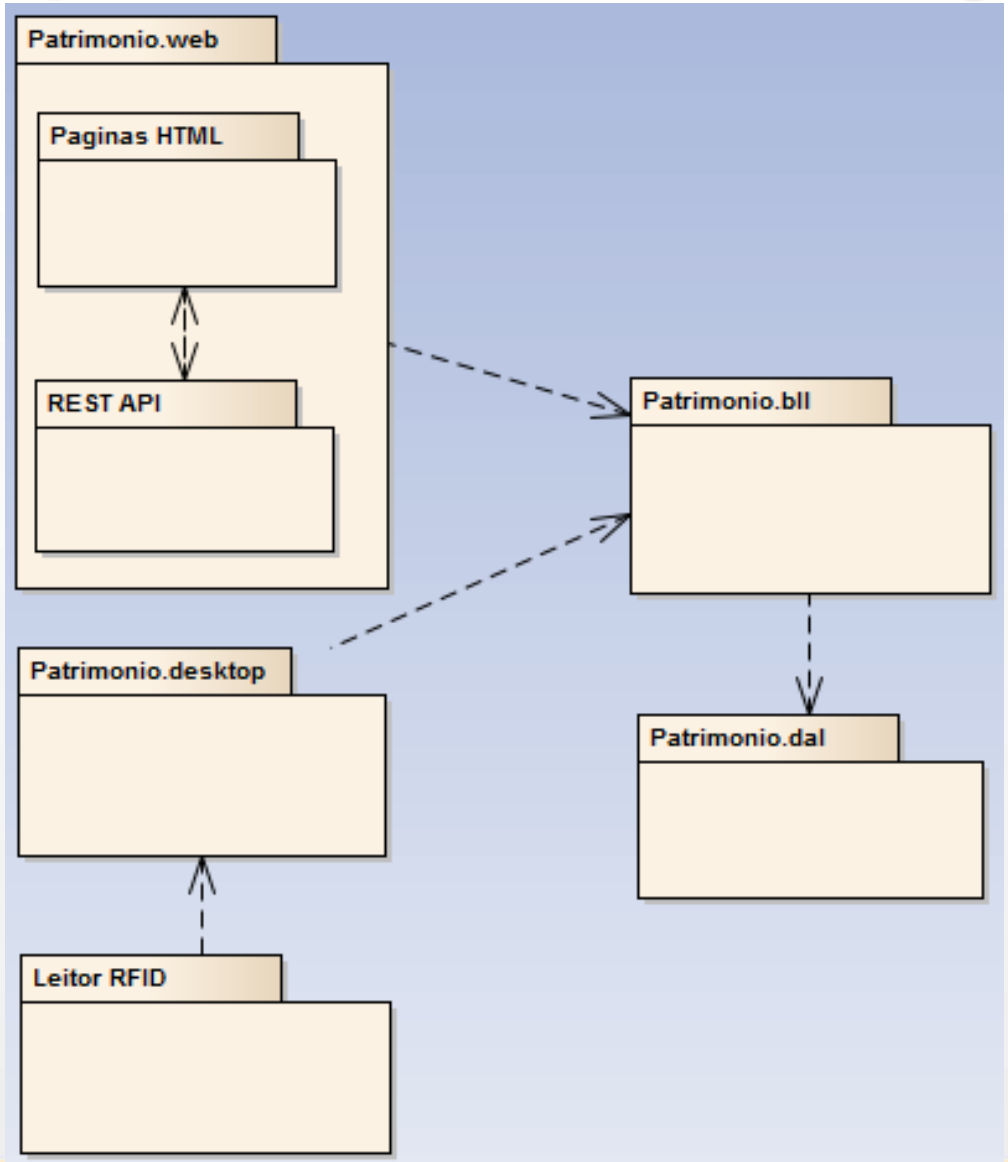
Desenvolvimento

Patrimônio Web

Patrimônio Desktop

Leitor RFID

Arquitetura da solução



Patrimonio.dal

```
namespace Patrimonio.dal
{
    using System;
    using System.Data.Entity;
    using System.Data.Entity.Infrastructure;

    public partial class PatrimonioContainer : DbContext
    {
        public PatrimonioContainer()
            : base("name=PatrimonioContainer")
        {
        }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            throw new UnintentionalCodeFirstException();
        }

        public virtual DbSet<Categoria> Categorias { get; set; }
        public virtual DbSet<Bem> Bens { get; set; }
        public virtual DbSet<Pessoa> Pessoas { get; set; }
        public virtual DbSet<Local> Locais { get; set; }
        public virtual DbSet<Departamento> Departamentos { get; set; }
        public virtual DbSet<LocalDepartamento> LocaisDepartamentos { get; set; }
        public virtual DbSet<Transferencia> Transferencias { get; set; }
        public virtual DbSet<Inventario> Inventarios { get; set; }
        public virtual DbSet<InventarioRealizado> InventariosRealizados { get; set; }
        public virtual DbSet<BensInventariados> BensInventariados { get; set; }
    }
}
```

Classe Transferencia

```
namespace Patrimonio.dal
{
    using System;
    using System.Collections.Generic;

    public partial class Transferencia
    {
        public int Id { get; set; }
        public System.DateTime DataSolicitacao { get; set; }
        public Nullable<System.DateTime> DataAprovacao { get; set; }
        public string Observacao { get; set; }
        public Nullable<bool> Aceito { get; set; }
        public string Aprovacao { get; set; }

        public virtual Bem Bem { get; set; }
        public virtual Usuario Solicitante { get; set; }
        public virtual Usuario Aprovador { get; set; }
        public virtual Departamento DepartamentoOrigem { get; set; }
        public virtual Departamento DepartamentoDestino { get; set; }
        public virtual Local LocalOrigem { get; set; }
        public virtual Local LocalDestino { get; set; }
    }
}
```

Módulo Patrimonio.bll

```
public static void RealizarConferencia(dal.Departamento dep, dal.Local loc)
{
    var inv = bll.Inventario.Ativo();

    // Registro do inventário realizado
    var realizado = new dal.InventarioRealizado()
    {
        Inventario = inv,
        Usuario = bll.Sistema.usuarioLogado,
        Departamento = dep,
        Local = loc,
        Data = DateTime.Now,
    };

    // Inclui os bens atuais do departamento e local no inventario realizado
    foreach (var bem in dep.Bens.Where(c => c.Local.Id == loc.Id))
    {
        realizado.BensInventariados.Add(new dal.BensInventariados()
        {
            InventarioRealizado = realizado,
            Bem = bem
        });
    }

    // Adiciona os dados coletados ao inventário atual e grava
    inv.InventarioRealizado.Add(realizado);
    Repositorio<dal.Inventario>.Commit();
}
```

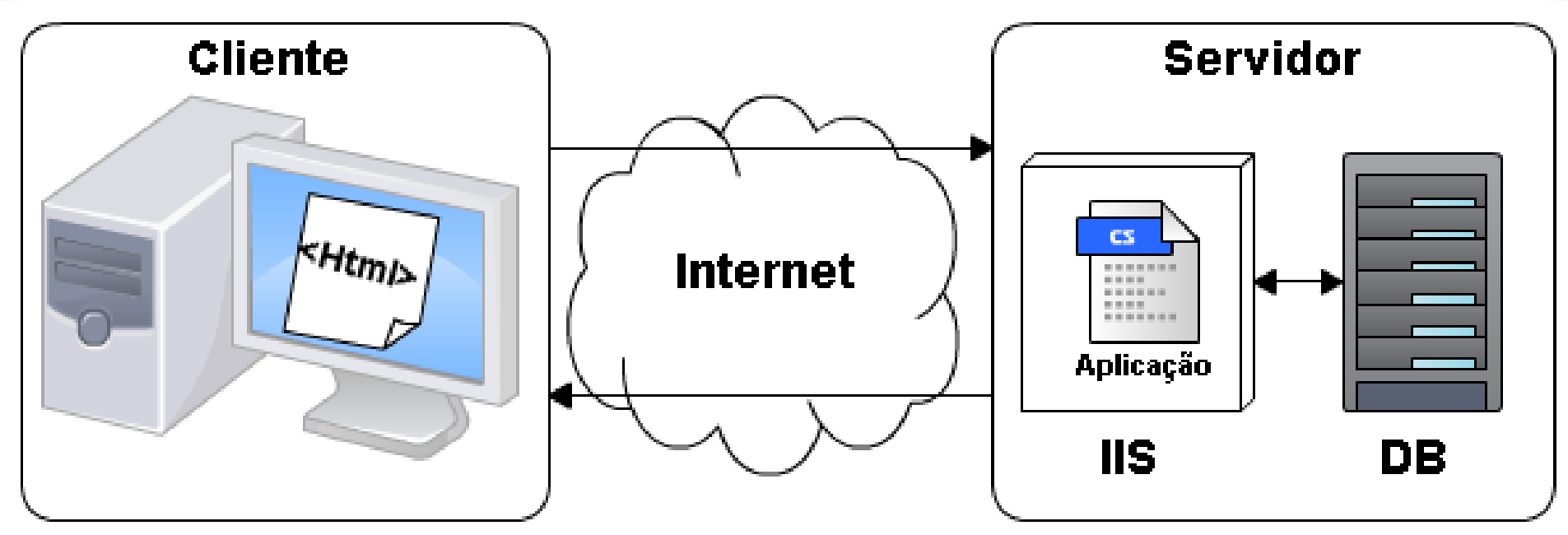
Classe Sistema

```
namespace Patrimonio.bll
{
    public class Sistema
    {
        private static PatrimonioContainer _context;
        public static PatrimonioContainer Context
        {
            get
            {
                if (_context == null)
                    _context = new PatrimonioContainer();

                return _context;
            }
            set { _context = value; }
        }

        private static dal.Usuario _UsuarioLogado;
        public static dal.Usuario usuarioLogado
        {
            get
            {
                return _UsuarioLogado;
            }
            set
            {
                _UsuarioLogado = value;
            }
        }
```

Classe Repositorio

```
public abstract class Repositorio<T> where T : class
{
    public static IQueryable<T> GetTodos()
    {
        return Sistema.Context.Set<T>();
    }

    public static IQueryable<T> Get(Expression<Func<T, bool>> predicate)
    {
        return Sistema.Context.Set<T>().Where(predicate);
    }

    public static T Find(params object[] key)...

    public static T First(Expression<Func<T, bool>> predicate)...

    public static void Adicionar(T entity)...

    public static void Atualizar(T entity)...

    public static void Deletar(Func<T, bool> predicate)
    {
        Sistema.Context.Set<T>()
       .Where(predicate).ToList()
       .ForEach(del => Sistema.Context.Set<T>().Remove(del));
    }

    public static void Commit()
    {
        Sistema.Context.SaveChanges();
    }
```

Módulo Patromonio.web



Página bens.html

```
<table class="table table-striped">
    <tr>
        <th><input type="checkbox" ng-model="allChecked" /></th>
        <th><a href="" ng-click="ordenacao('Categoria')">Categoria</a></th>
        <th><a href="" ng-click="ordenacao('Nome')">Nome</a></th>
        <th><a href="" ng-click="ordenacao('Plaqueta')">Plaqueta</a></th>
        <th><a href="" ng-click="ordenacao('TagUID')">TagUID</a></th>
    </tr>
    <tr ng-repeat="bem in filteredBens | orderBy: criterioOrderBy : direcaoOrderBy">
        <td><input type="checkbox" ng-model="bem.Selecionado"/></td>
        <td>{{bem.categoria.Nome}}</td>
        <td>{{bem.Nome}}</td>
        <td>{{bem.Plaqueta}}</td>
        <td>{{bem.TagUID}}</td>
    </tr>
</table>

<div class="container text-center">
    <button class="btn btn-primary" ng-click="transferir()">Transferir</button>
</div>
```

BensCtrl.js

```
app
.controller('bensCtrl', function ($scope, $rootScope, $location, bensService, departamentoService
            , localService, pessoaService, categoriaService) {
    $scope.departamentos = [];
    $scope.locais = [];
    $scope.bens = [];
    $scope.responsaveis = [];
    $scope.categorias = [];

    $scope.depFilter = '';
    $scope.locFilter = '';
    $scope.filteredBens = [];

    // Inicializa tela carregando Bens do usuario logado
    bensService.getBens()
        .then(function (response) {
            $scope.bens = response.data;
            $scope.filteredBens = $scope.bens;

            // Carrega os departamentos
            departamentoService.getDepartamentos()
                .then(function (response) {
                    $scope.departamentos = response.data;

                    localService.getLocais()
                        .then(function (response) {
                            $scope.locais = response.data;

                            // Carrega Categorias
                            categoriaService.getAll()
                                .then(function (response) {
```

API do serviço de bens

```
public class BensController : ApiController
{
    public JsonResult<List<Models.Bem>> Get(int idDepartamento, int idLocal)...

    [HttpPost]
    public JsonResult<Models.Response> SaveBem([FromBody] Models.Bem bem)
    {
        try
        {
            int idCategoria = bem.categoria != null ? bem.categoria.Id : 0;
            int idDepartamento = bem.departamento != null ? bem.departamento.Id : 0;
            int idLocal = bem.local != null ? bem.local.Id : 0;
            int idResponsavel = bem.responsavel != null ? bem.responsavel.Id : 0;

            bll.Bem.Alterar(
                bem.Id,
                bem.Nome,
                bem.Plaqueta,
                bem.TagUID,
                idCategoria,
                idDepartamento,
                idLocal,
                idResponsavel);

            return Json(new Models.Response() { Sucesso = true });
        }
        catch (Exception ex)
        {
            return Json(new Models.Response() { Sucesso = false, Mensagem = ex.Message });
        }
    }
```

Demonstração

- Cenário 1: Cadastros básicos; Identificação de Bem;
- Cenário 2: Movimentação de bens;
- Cenário 3: Inventario dos bens movimentados;
- Cenário 4: Inventário com bens movimentados mas não transferidos;

Resultados e Discussões

- Identificação dos bens patrimoniais auxiliando o processo de inventário patrimonial;
- Levantamento do inventário patrimonial sem a necessidade de conferência de plaquetas por identificação visual;
- Sincronização dos dados eliminando erros de conferência.

Semelhanças e Diferenças

- Santos (2001):
- Auxílio no gerenciamento de informações da cadeia produtiva de bovinos;
- Leitor Olimex SCL05 - 128 kHz;

Semelhanças e Diferenças

- Zendin (2005):
- RFID para controle de estoque;
- Leitor ZEBEK Z3070 - 13.56 MHz

Semelhanças e Diferenças

- Diener (2002):
- Tecnologia de rádio frequência para identificação dos veículos;
- Módulo PICLAB 4A – 315 MHz;

Conclusões

- Expectativa e Objetivos atendidos;
- Modulo RC522 possibilitou a identificação das *tags*;
- O ambiente de desenvolvimento e frameworks escolhidos possibilitaram rápido desenvolvimento e validação da solução proposta;
- Destaque para o baixo custo dos componentes.

LIMITAÇÕES

- EEPROM de 1Kb;
- 4 bytes por tag;
- 256 tags;
- 64 estações de trabalho.

EXTENSÕES

Desenvolver o sistema de inventário para dispositivos mobile.

Unificar os três módulos.

Utilizar o sensor NFC para leitura das tags.