|  |
| --- |
| **人事情報Webシステム仕様書** |

**目次**

[1 システムの目的 3](#_Toc73068773)

[2 システム構成 3](#_Toc73068774)

[3 ユースケース 4](#_Toc73068775)

[4 業務フロー 5](#_Toc73068776)

[5 ユースケースシナリオ 5](#_Toc73068777)

[5.1 一般社員の場合 5](#_Toc73068778)

[5.2 人事担当者の場合 5](#_Toc73068779)

[6 画面 6](#_Toc73068780)

[7 DB 仕様 8](#_Toc73068781)

[8 Web API 仕様 9](#_Toc73068782)

[9 クライアント側実装技術仕様 9](#_Toc73068783)

[10 その他 9](#_Toc73068784)

# システムの目的

現在、社員の情報が記録されているデータベースが存在している。

以前はDBを直接メンテナンスしていた。社員情報をメンテナンスするには、

一般社員には見せたくない人事情報に触れてしまうため、

人事が直接メンテナンスを行いたいがDBの知識が必須であった。

そのため以下の目的を満たすシステムを構築する。

・DBの知識の無い人事がWeb上で管理できるようにしたい。

・一般社員も閲覧可能な情報を確認できるようにしたい。

・社員ではない人には見せたくない。

# システム構成

Visual Studio のソリューションのASP.NET Web API

ソリューション名：PersonnelInformationWebAPI.sln

プロジェクト名　：PersonnelInformationWebAPI

Webサーバーは開発で使用する　IIS Expressをそのまま利用。

サーバー名：localhost、ポート番号：４桁以上の数値とする。(5桁推奨、自動値で良い)

DBは既存のものをそのまま利用してもよい、既存のDBとデータを元にして

システムに取り込んでもよい。

既存のDBは

C:\testDB\AvaNyadeEmployee.mdf

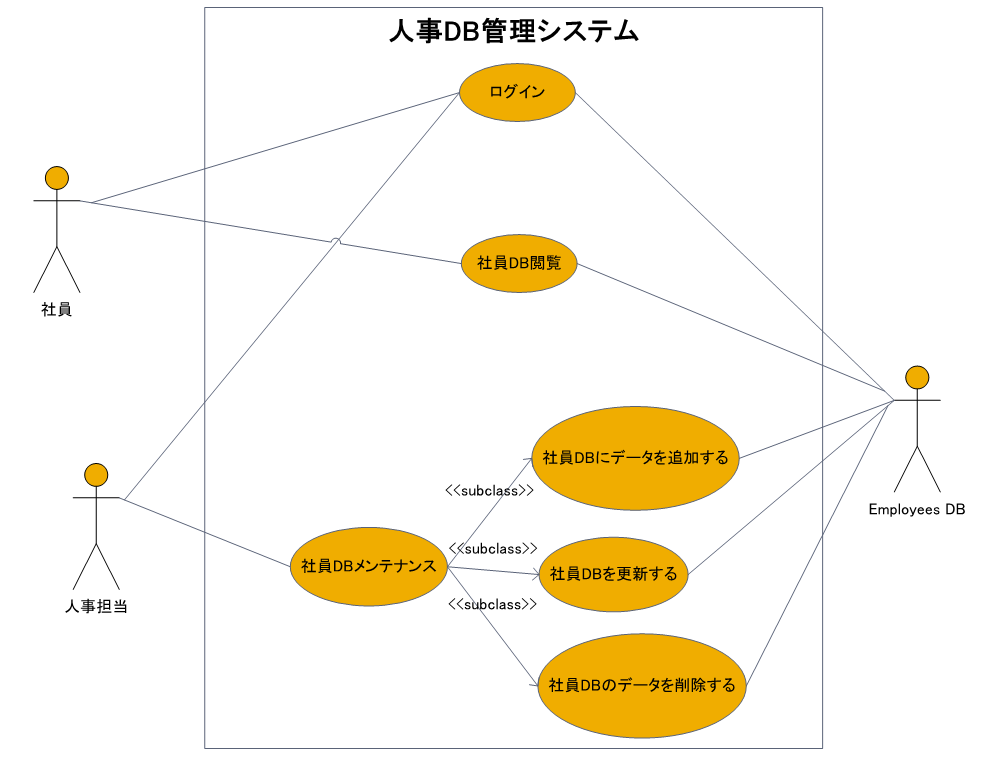
に配置する。

|  |
| --- |
| クライアント Webサーバー(IIS/local) DB(local) |
| JavaScripr/jQuery/ajax ASP.NET Web API T-SQL |

# ユースケース

本システムのユースケースを以下に示す。

非機能要件について、今回は考慮しない。



ユースケースの説明

利用者：一般社員、人事。人事はHRという名称のロールを設定

社員が人事であるという判断は既存のDBに登録されている情報をそのまま利用

DB ：AvaNyadeEmployee

DBの情報を変更した場合は、変更した理由と説明を書いてください

既存DB利用可、既存のものをベースに新規作成可。

社員のテーブルと認証用のテーブルは分けてもよい

ユースケース図に関して不明点がある場合は、以下で調べること。  
オブジェクトクラブ：UML入門  
<http://objectclub.jp/technicaldoc/uml/umlintro>  
特に、２章部分を参照：<http://objectclub.jp/technicaldoc/uml/umlintro2>

# 業務フロー

（ユースケースシナリオに基づくため）割愛

# ユースケースシナリオ

## 一般社員の場合

一般社員は、Webページアクセスする

<http://localhost:7777/>Login.html (ポート番号は４桁以上の任意の数値)

ログインページが表示される

自分のメールアドレスとパスワードでログインを行う

パスワードは P@ssw0rd1!

　　　ログイン後は一覧が表示できるEmployeesListページに遷移

<http://localhost:7777/EmployeesList.html>

Employees DBの以下のカラムのみ表示される。

EmployeeID, EmployeeName, Email, HireFiscalYear

EmployeeIDの昇順に表示。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DBカラム名 | EmployeeID | EmployeeName | Email | HireFiscalYear |
| Web表示名 | 社員Id | 社員名 | メール | 雇用年度 |

## 人事担当者の場合

基本は社員の場合と同じ

人事担当者がログイン後は一覧が表示できるIndexページに遷移

[http://localhost:7777/](http://localhost:7777/HR/Employees/Index)HREmployeeMaintenance.html

Employeesテーブルと、BelongToテーブルと、Departmentsテーブルを連結して、

現在在籍している社員の全内容と部署名が認識できるDBのすべてのカラムが表示される。カラム名は日本語で表示。重複するカラムの表示は不要。EmployeeIDの昇順に表示。

DBメンテナンス用にCRUDすべての処理と、詳細（Detail）のページを作成

　　 　　メンテナンスはEmployeesテーブルのみで良い。

（Detailの表示対象はEmployeesテーブルのみ）

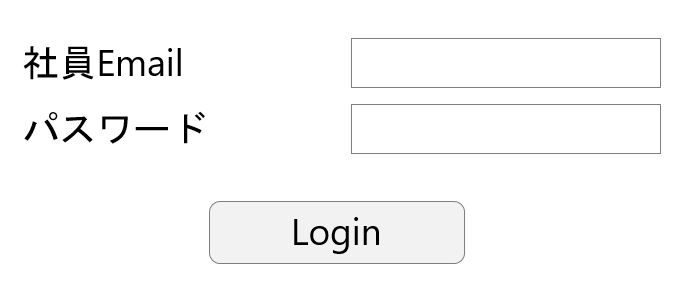
# 画面

画面イメージの例（イメージと一致しているところまでは問わない。機能、出力結果を重視）

ログイン画面

画面名：Login.html

画面イメージ



制約事項

社員 Email、パスワード は必須入力

社員 Email はメール形式以外の値はエラー　（例：間に必ず @ が含まれる等）

パスワードはセキュリティの関係で、入力した文字を ●　で表示

**オプション**

新規登録、パスワード忘れ画面と機能

一般社員が閲覧する画面

画面名：EmployeesList.html



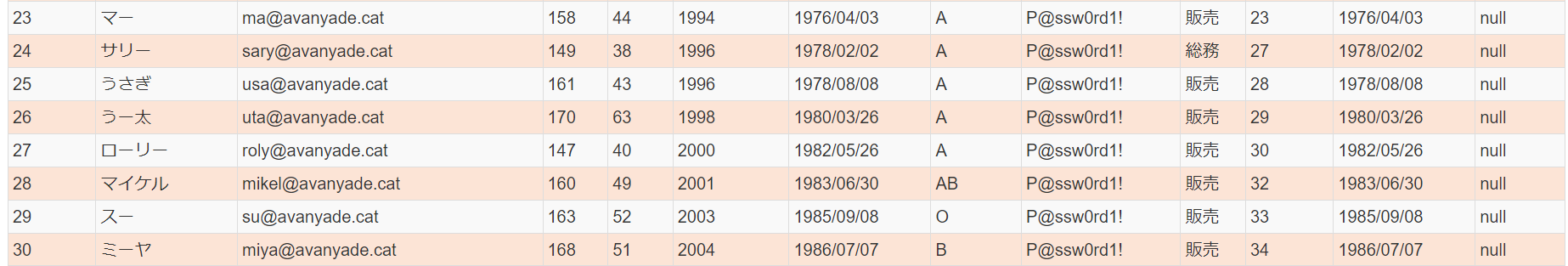
**オプション**

　　　　ソート処理。カラム名クリックで結果がソートされる

人事担当者が閲覧する画面

画面名：HREmployeeMaintenance.html

テーブル

中程度の精度で自動的に生成された説明  


制約事項

DBメンテナンス用にCRUDすべての処理と、詳細（Detail）のページがあるため

詳細（Detail）のページへの遷移処理

　　 　　メンテナンスはEmployeesテーブルのみで良い。

（Detailの表示対象はEmployeesテーブルのみ）

**オプション**

検索処理

一覧の上部に検索用のテキスト入力エリアと、ボタンを配置し検索可能にする

検索項目は社員名で部分一致できること

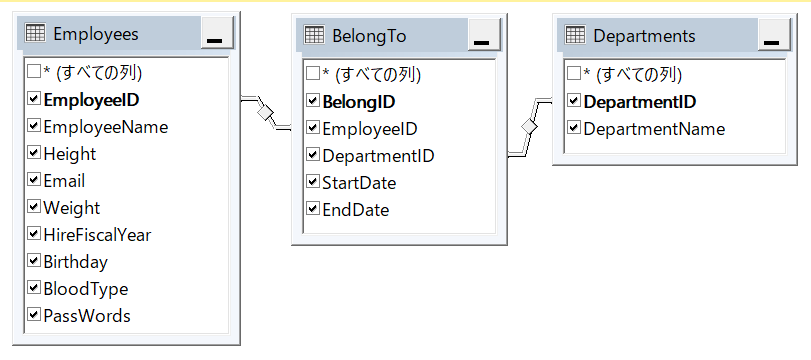
（例「ー」で検索 →シマゴロー、ミチロー、うー太、ミーヤ等がヒット）

　　　　ソート処理。カラム名クリックで結果がソートされる

# DB 仕様

各種テーブルは以下のような関係である。

人事は、Departmentsテーブルの値が「人事」となっている社員



Employeesテーブル

|  |  |  |  |
| --- | --- | --- | --- |
| **カラム名** | **型** | **NULL許可** | **意味** |
| EmployeeID | INTEGER | NOT NULL, | 従業員ID（主キー） |
| EmployeeName | VARCHAR(100) |  | 従業員名 |
| Height | INTEGER |  | 身長(cm) |
| EMail | VARCHAR(100) |  | メールアドレス |
| Weight | INTEGER |  | 体重(kg) |
| HireFiscalYear | INTEGER |  | 入社年度 |
| Birthday | DATETIME |  | 誕生日 |
| BloodType | VARCHAR(2) |  | 血液型 |
| PassWords | VARCHAR(10) |  | パスワード |

BelongTo テーブル

|  |  |  |  |
| --- | --- | --- | --- |
| **カラム名** | **型** | **NULL許可** | **意味** |
| BelongID | INTEGER | NOT NULL, | 所属ID |
| EmployeeID | INTEGER |  | 従業員ID |
| DepartmentID | INTEGER |  | 部署ID |
| StartDate | DATETIME |  | 入社日 |
| EndDate | DATETIME |  | 退職日 |

Departments

|  |  |  |  |
| --- | --- | --- | --- |
| **カラム名** | **型** | **NULL許可** | **意味** |
| DepartmentID | INTEGER | NOT NULL, | 部署ID |
| DepartmentName | VARCHAR(100) |  | 部署名 |

# Web API 仕様

Web API に関しては必要に応じていくつ作成しても良い。  
認証に関しては、簡単な独自の方式で良い。（OAuthの実装を含むASP.NET Identityを実装する必要はない）

認証方法の例として、ログインの情報をDBに問い合わせて、部署名で判断する仕組みで良い。

（人事のアクセス権に関して、OAuthの実装のASP.NET Identityのロールまでは使用しなくても良い。）

（ログインせず直接APIを操作できても構わない）

# クライアント側実装技術仕様

クライアント側は 画面を実装する技術として、html/css/JavaScript jQuery ajaxを推奨しますが、実装方法はお任せします。ソリューションのフォルダの構成、ファイル名などは自由に実装してかまいません。

（本来の仕様書では明示的に定義するが、今回はテストの為自由な実装となります。）

# その他

成果物に関して、説明が必要であれば、こちらに記載し、提出物に含めてください。

―　以上 －