David Ziegelbauer

ID: 000496966

Design Document

D424 Task 3

Table of Contents

[Database Design 2](#_Toc157243999)

[Entities 2](#_Toc157244000)

[Ticket 2](#_Toc157244001)

[Customer 4](#_Toc157244002)

[User 4](#_Toc157244003)

[Entity Relationship Diagram 8](#_Toc157244004)

[Application Design 9](#_Toc157244005)

[Tiered Architecture 9](#_Toc157244006)

[Web Tier 9](#_Toc157244007)

[Business tier 9](#_Toc157244008)

[Persistence Tier 9](#_Toc157244009)

[Object UML 10](#_Toc157244010)

# Database Design

There are two major entities to model for this application, both of which required the design of other entities to support them. The first, and most obvious, is a ticket. This was done from scratch to the exact application specifications. It was also necessary to have a user entity; but it more sense to extend the user entity from ASP.NET Identity library for this purpose. Utilizing a well-tested, production ready identity library enhances the security posture of the application and avoids exposing critical corporate data to unnecessary risk.

## Entities

### Ticket

The ticket entity represents a work ticket. It maintains references to who created it, modified it last, and who it is assigned to. It has a title and narrative description of the work performed. It maintains a reference to the customer for whom the work is performed. It is created using the following DDL:

CREATE TABLE IF NOT EXISTS public."Tickets"

(

"Id" integer NOT NULL GENERATED BY DEFAULT AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

"Title" character varying(50) COLLATE pg\_catalog."default" NOT NULL,

"Description" text COLLATE pg\_catalog."default" NOT NULL,

"CreatorId" text COLLATE pg\_catalog."default",

"ModifierId" text COLLATE pg\_catalog."default",

"AssignedToId" text COLLATE pg\_catalog."default",

"CustomerId" integer NOT NULL,

"Timestamp" bytea,

"Created" timestamp without time zone NOT NULL,

"Modified" timestamp without time zone NOT NULL,

"Closed" timestamp without time zone,

"Status" integer NOT NULL,

"BillableHours" integer NOT NULL,

CONSTRAINT "PK\_Tickets" PRIMARY KEY ("Id")

);

ALTER TABLE IF EXISTS public."Tickets"

ADD CONSTRAINT "FK\_Tickets\_AspNetUsers\_AssignedToId" FOREIGN KEY ("AssignedToId")

REFERENCES public."AspNetUsers" ("Id") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

CREATE INDEX IF NOT EXISTS "IX\_Tickets\_AssignedToId"

ON public."Tickets"("AssignedToId");

ALTER TABLE IF EXISTS public."Tickets"

ADD CONSTRAINT "FK\_Tickets\_AspNetUsers\_CreatorId" FOREIGN KEY ("CreatorId")

REFERENCES public."AspNetUsers" ("Id") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

CREATE INDEX IF NOT EXISTS "IX\_Tickets\_CreatorId"

ON public."Tickets"("CreatorId");

ALTER TABLE IF EXISTS public."Tickets"

ADD CONSTRAINT "FK\_Tickets\_AspNetUsers\_ModifierId" FOREIGN KEY ("ModifierId")

REFERENCES public."AspNetUsers" ("Id") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

CREATE INDEX IF NOT EXISTS "IX\_Tickets\_ModifierId"

ON public."Tickets"("ModifierId");

ALTER TABLE IF EXISTS public."Tickets"

ADD CONSTRAINT "FK\_Tickets\_Customers\_CustomerId" FOREIGN KEY ("CustomerId")

REFERENCES public."Customers" ("Id") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE;

CREATE INDEX IF NOT EXISTS "IX\_Tickets\_CustomerId"

ON public."Tickets"("CustomerId");

### Customer

The customer is a simple reference to a client. It includes the client’s name and contact information. It is created using the following DDL:

CREATE TABLE IF NOT EXISTS public."Customers"

(

"Id" integer NOT NULL GENERATED BY DEFAULT AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

"Name" text COLLATE pg\_catalog."default",

"StreetAddress" text COLLATE pg\_catalog."default",

"City" text COLLATE pg\_catalog."default",

"State" text COLLATE pg\_catalog."default",

"Zip" text COLLATE pg\_catalog."default",

CONSTRAINT "PK\_Customers" PRIMARY KEY ("Id")

);

### User

This entity represents individual employees. Most of this entity is not designed specifically for this application but is built to support the ASP.NET Identity library. This application extends the default object entity by adding a FirstName and LastName field to display user identities in a more friendly format. A ChargeableRate field is also added to calculate how much to charge customers for each employee’s work. It is created with the following DDL:

CREATE TABLE IF NOT EXISTS public."AspNetRoleClaims"

(

"Id" integer NOT NULL GENERATED BY DEFAULT AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

"RoleId" text COLLATE pg\_catalog."default" NOT NULL,

"ClaimType" text COLLATE pg\_catalog."default",

"ClaimValue" text COLLATE pg\_catalog."default",

CONSTRAINT "PK\_AspNetRoleClaims" PRIMARY KEY ("Id")

);

CREATE TABLE IF NOT EXISTS public."AspNetRoles"

(

"Id" text COLLATE pg\_catalog."default" NOT NULL,

"Name" character varying(256) COLLATE pg\_catalog."default",

"NormalizedName" character varying(256) COLLATE pg\_catalog."default",

"ConcurrencyStamp" text COLLATE pg\_catalog."default",

CONSTRAINT "PK\_AspNetRoles" PRIMARY KEY ("Id")

);

CREATE TABLE IF NOT EXISTS public."AspNetUserClaims"

(

"Id" integer NOT NULL GENERATED BY DEFAULT AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

"UserId" text COLLATE pg\_catalog."default" NOT NULL,

"ClaimType" text COLLATE pg\_catalog."default",

"ClaimValue" text COLLATE pg\_catalog."default",

CONSTRAINT "PK\_AspNetUserClaims" PRIMARY KEY ("Id")

);

CREATE TABLE IF NOT EXISTS public."AspNetUserLogins"

(

"LoginProvider" text COLLATE pg\_catalog."default" NOT NULL,

"ProviderKey" text COLLATE pg\_catalog."default" NOT NULL,

"ProviderDisplayName" text COLLATE pg\_catalog."default",

"UserId" text COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "PK\_AspNetUserLogins" PRIMARY KEY ("LoginProvider", "ProviderKey")

);

CREATE TABLE IF NOT EXISTS public."AspNetUserRoles"

(

"UserId" text COLLATE pg\_catalog."default" NOT NULL,

"RoleId" text COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "PK\_AspNetUserRoles" PRIMARY KEY ("UserId", "RoleId")

);

CREATE TABLE IF NOT EXISTS public."AspNetUserTokens"

(

"UserId" text COLLATE pg\_catalog."default" NOT NULL,

"LoginProvider" text COLLATE pg\_catalog."default" NOT NULL,

"Name" text COLLATE pg\_catalog."default" NOT NULL,

"Value" text COLLATE pg\_catalog."default",

CONSTRAINT "PK\_AspNetUserTokens" PRIMARY KEY ("UserId", "LoginProvider", "Name")

);

CREATE TABLE IF NOT EXISTS public."AspNetUsers"

(

"Id" text COLLATE pg\_catalog."default" NOT NULL,

"ChargeableRate" numeric NOT NULL,

"FirstName" character varying(50) COLLATE pg\_catalog."default",

"LastName" character varying(50) COLLATE pg\_catalog."default",

"UserName" character varying(256) COLLATE pg\_catalog."default",

"NormalizedUserName" character varying(256) COLLATE pg\_catalog."default",

"Email" character varying(256) COLLATE pg\_catalog."default",

"NormalizedEmail" character varying(256) COLLATE pg\_catalog."default",

"EmailConfirmed" boolean NOT NULL,

"PasswordHash" text COLLATE pg\_catalog."default",

"SecurityStamp" text COLLATE pg\_catalog."default",

"ConcurrencyStamp" text COLLATE pg\_catalog."default",

"PhoneNumber" text COLLATE pg\_catalog."default",

"PhoneNumberConfirmed" boolean NOT NULL,

"TwoFactorEnabled" boolean NOT NULL,

"LockoutEnd" timestamp with time zone,

"LockoutEnabled" boolean NOT NULL,

"AccessFailedCount" integer NOT NULL,

CONSTRAINT "PK\_AspNetUsers" PRIMARY KEY ("Id")

);

ALTER TABLE IF EXISTS public."AspNetRoleClaims"

ADD CONSTRAINT "FK\_AspNetRoleClaims\_AspNetRoles\_RoleId" FOREIGN KEY ("RoleId")

REFERENCES public."AspNetRoles" ("Id") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE;

CREATE INDEX IF NOT EXISTS "IX\_AspNetRoleClaims\_RoleId"

ON public."AspNetRoleClaims"("RoleId");

ALTER TABLE IF EXISTS public."AspNetUserClaims"

ADD CONSTRAINT "FK\_AspNetUserClaims\_AspNetUsers\_UserId" FOREIGN KEY ("UserId")

REFERENCES public."AspNetUsers" ("Id") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE;

CREATE INDEX IF NOT EXISTS "IX\_AspNetUserClaims\_UserId"

ON public."AspNetUserClaims"("UserId");

ALTER TABLE IF EXISTS public."AspNetUserLogins"

ADD CONSTRAINT "FK\_AspNetUserLogins\_AspNetUsers\_UserId" FOREIGN KEY ("UserId")

REFERENCES public."AspNetUsers" ("Id") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE;

CREATE INDEX IF NOT EXISTS "IX\_AspNetUserLogins\_UserId"

ON public."AspNetUserLogins"("UserId");

ALTER TABLE IF EXISTS public."AspNetUserRoles"

ADD CONSTRAINT "FK\_AspNetUserRoles\_AspNetRoles\_RoleId" FOREIGN KEY ("RoleId")

REFERENCES public."AspNetRoles" ("Id") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE;

CREATE INDEX IF NOT EXISTS "IX\_AspNetUserRoles\_RoleId"

ON public."AspNetUserRoles"("RoleId");

ALTER TABLE IF EXISTS public."AspNetUserRoles"

ADD CONSTRAINT "FK\_AspNetUserRoles\_AspNetUsers\_UserId" FOREIGN KEY ("UserId")

REFERENCES public."AspNetUsers" ("Id") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE;

ALTER TABLE IF EXISTS public."AspNetUserTokens"

ADD CONSTRAINT "FK\_AspNetUserTokens\_AspNetUsers\_UserId" FOREIGN KEY ("UserId")

REFERENCES public."AspNetUsers" ("Id") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE;

## Entity Relationship Diagram

For the sake of brevity, default entities needed for ASP.NET Identity are not included in this diagram. This depicts entities custom to, or modified for, the TicketPro application.

A screenshot of a computer

Description automatically generated

Figure 1: Entity Relationship Diagram

# Application Design

## Tiered Architecture

The TicketPro application is built in three tiers. Each tier is responsible for a different aspect of the application’s function.

A diagram of a computer

Description automatically generated

Figure 2: TicketPro Architecture

### Web Tier

The web tier is responsible for interaction with the user. It renders data provided by business tier into text or graphics that the user can consume. It translates user input into structured data for processing by the business tier. It also performs validation of user input to ensure it meets the requirements of the business layer.

### Business tier

The business tier is responsible for applying business rules to data supplied by the web tier or retrieved from the persistence tier. It performs computations and transformations on data to build reports. It enforces rules for what may happen to some data values based on other values.

### Persistence Tier

The persistence tier is responsible for receiving data from the business tier, storing it, and retrieving it when requested by the business tier.

## Object UML

### Web Tier

A screenshot of a computer screen

Description automatically generated

Figure 3: Web Tier Data Transfer Objects

A screenshot of a computer

Description automatically generated

Figure 4: Web Tier UI Components

### Business Tier

A screenshot of a program

Description automatically generated

Figure 5: Business Tier Components

### Persistence Tier

A diagram of a software application

Description automatically generated with medium confidence

Figure 6: Persistence Tier Entity Objects