QMF Split - Initial domain and NHibernate changes

Description

Task 1 - Update properties of QMF domain object

The first task is to add some missing properties to the QMF domain object (**Vision.Api.DotNet.Domain.Qmfs.Qmf.cs**)
The properties to add are as follows...

- RetentionPercent (int?)
- PointsPerUnitSold (int?)

Task 2 - Create new LabelBranding domain object

We also need to create a new Domain object for a database table that doesn't currently exist in Vision.

This will be: **Vision.Api.DotNet.Domain.Qmf.LabelBranding.cs**

(For this, you can copy and paste the existing **BarStandard.cs** class to create the LabelBranding class, then just change the name and properties).

The LabelBranding class will inherit from Entity (just as LabelBranding.cs does), and it will have the following properties.

- Name (string)
- Website (string)
- Email (string)
- QrCodePrefix (string)
- BrandLogoFilePath (string)
- ServiceTagAdditionalText (string)
- DisplayElectricalWarnings (bool?)

Task 3 - Add NHibernate mappings for the new QMF properties

For the fields added in task 1, we'll also have to provide mappings for NHibernate so it knows how to retrieve and store the data in the database.

For this we'll need to edit the following class:
Vision.Api.DotNet.Domain.NHibernate.Mapping.Qmf.QmfMapping.cs

The fields will use the standard Property mapping:

Property(x => x.*{PropertyName}***, map => map.Column("***{DatabaseColumn}***"));**

They'll map as follows.

- **RetentionPercent** will map to the database column: **qmf_retention_percent**
- **PointsPerUnitSold** will map to the database column: **PointsPerUnitSold**

Task 4 - Add NHibernate mappings for the new LabelBranding class

We'll also need to add mappings for our new **LabelBranding** object.
However this doesn't actually have a class yet, so we'll have to add one.

(For this, you can copy and paste the following class, and again just change the name and properties).
Copy the class:
Vision.Api.DotNet.Domain.NHibernate.Mapping.Qmf.BarStandardMapping.cs
To then create the new class:
Vision.Api.DotNet.Domain.NHibernate.Mapping.Qmf.LabelBrandingMapping.cs

The Table will map to: **LabelBranding**

The Id will map exactly the same as in **BarStandardMapping**:

```
Id(x => x.Id, map =>
{
map.Column("Id");
map.Generator(Generators.GuidComb);
});
```

The other fields will use the standard Property mapping

Property(x => x.{PropertyName}, map => map.Column("{DatabaseColumn}"));

They'll map as follows.

- Name will map to the database column: Name
- Website will map to the database column: Website
- Email will map to the database column: Email
- QrCodePrefix will map to the database column: QrCodePrefix
- BrandLogoFilePath will map to the database column: BrandLogoFilePath
- ServiceTagAdditionalText will map to the database column: ServiceTagAdditionalText

- DisplayElectricalWarnings will map to the database column: DisplayElectricalWarnings

The final step here will be to register our new LabelBrandingMapping.cs class in the main Mappings class.
This can be found at: **Vision.Api.DotNet.Domain.NHibernate.Mapping.Mapping.cs**

For this, you just need to add a new mapping line in like the other entries in the class.

Task 5 - Add LabelBranding property to QMF domain object

Now that our new LabelBranding object exists, we can add it as property to the Qmf domain object.
Back in our Vision.Api.DotNet.Domain.Qmfs.Qmf.cs class, we now need to add the property:

- LabelBranding (LabelBranding)
  (in other words, the property called LabelBranding, has a type of 'LabelBranding' which is the domain object you just created).

Task 6 - Add NHibernate mapping for the new LabelBranding property in the Qmf object

Finally, we'll be editing the following class again:
Vision.Api.DotNet.Domain.NHibernate.Mapping.Qmf.QmfMapping.cs

And here we add in a mapping for LabelBranding.
Unlike the other properties we added though, this does not use a standard 'Property' mapping, but instead it uses a 'ManyToOne' mapping, meaning this is a foreign key link to another complex object, instead of something simple like a string or int.
The ManyToOne mapping has the syntax:

```
ManyToOne(x => x.{PropertyName}, map => map.Column("{DatabaseColumn}"));
```

The ManyToOne mapping will be as follows.

- LabelBranding will map to the database column: LabelBranding_id