ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής

Σύγχρονα Θέματα Τεχνολογίας Λογισμικού - Εργασία στο Μοντέλο MVC - Τεχνικό Εγχειρίδιο

Στοιχεία Φοιτητών	Σμύρης Βασίλειος - Π16131
	Πετριδίσογλου Παναγιώτης - Π16115
	Παπαδόπουλος Αθανάσιος - Π16108
Ημερομηνία παράδοσης	16/02/2020

messe

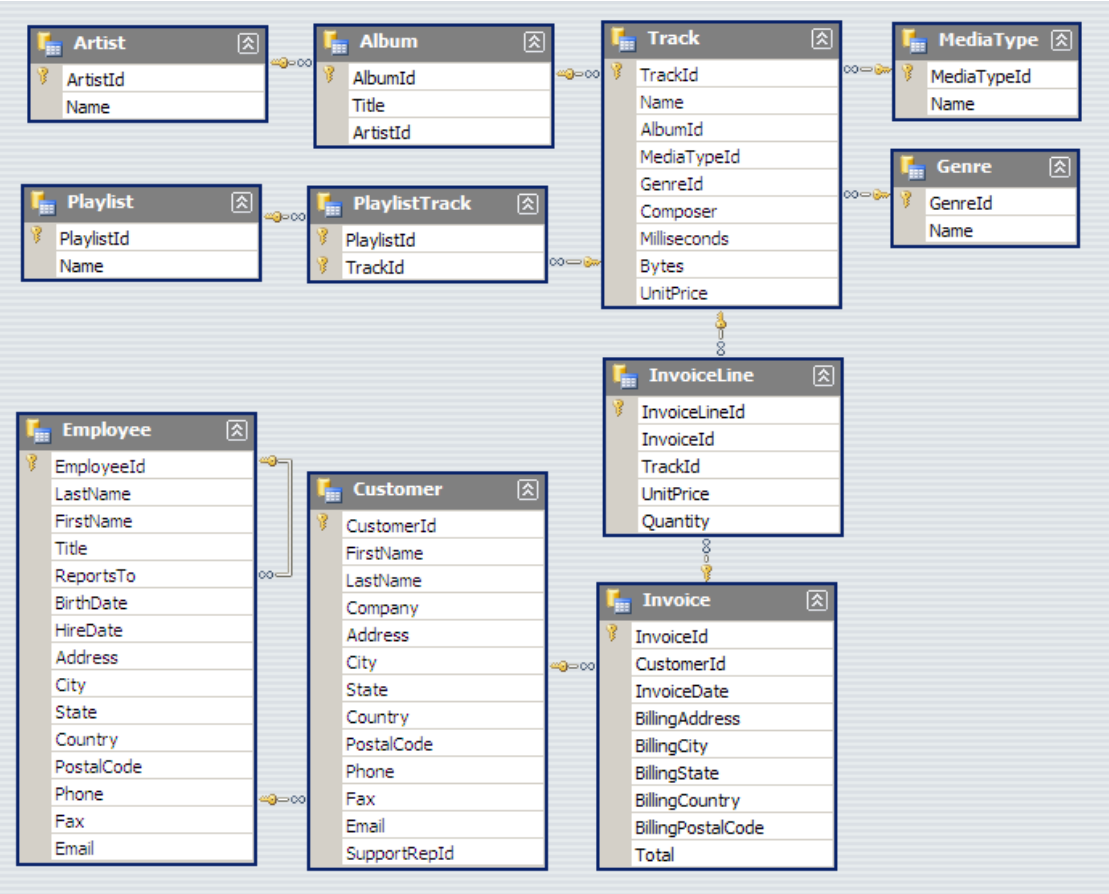
ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1 Εισαγωγή

Στο παρόν τεχνικό εγχειρίδιο, αναλύεται ο τρόπος με τον οποίο χρησιμοποιούνται τα διάφορα αντικείμενα της βάσης δεδομένων από το μοντέλο MVC, μέσα στην εφαρμογή του ηλεκτρονικού δισκοπωλείου.

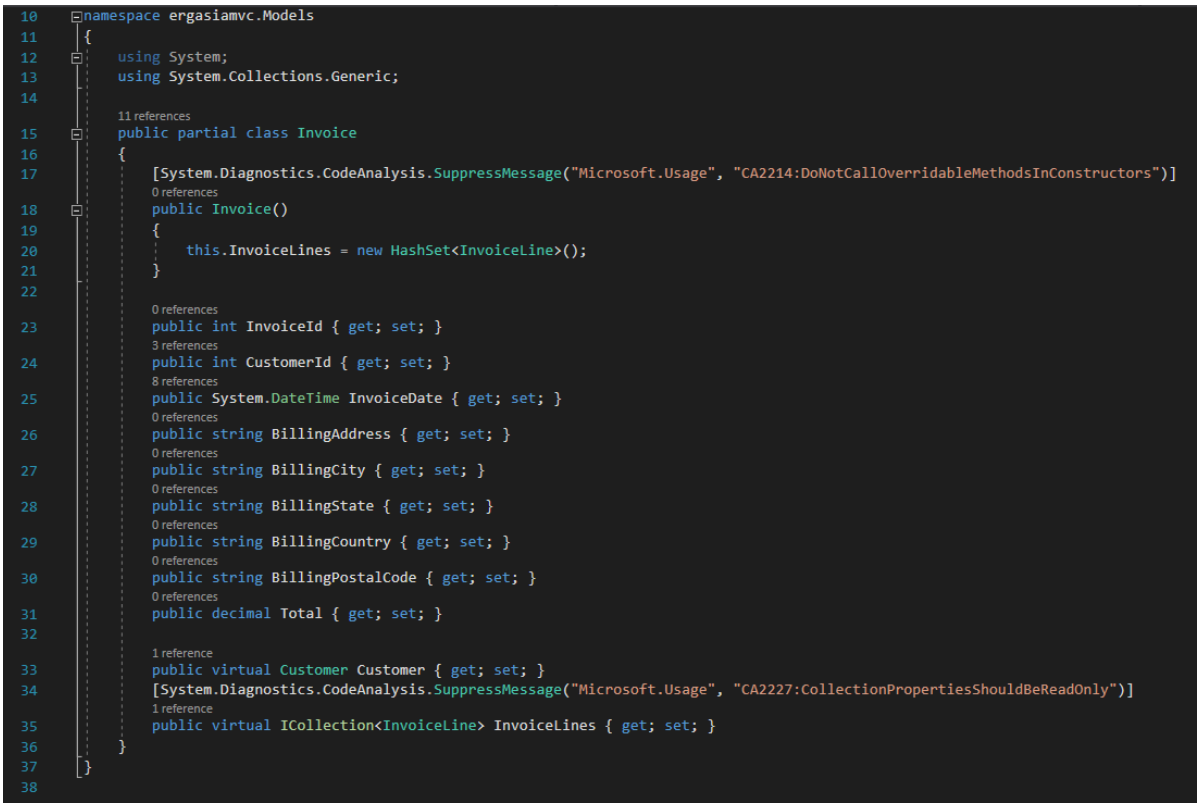
2 Models

Όπως φαίνεται στην Εικόνα 1, η βάση δεδομένων με την οποία συνδέεται η εφαρμογή ηλεκτρονικού δισκοπωλείου, περιέχει 11 πίνακες. Οι 10 από αυτούς τους πίνακες, οι Artist, Album, Track, MediaType, Genre, Playlist, InvoiceLine, Invoice, Customer και Employee, αντιπροσωπεύουν διακριτά αντικείμενα, κάποια από τα οποία έχουν μεταξύ τους σχέσεις ένα προς πολλά. Ο πίνακας PlaylistTrack είναι ιδιαίτερος, καθώς σε αντίθεση με τους άλλους αντιπροσωπεύει σχέση πολλά προς πολλά και όλες του οι στήλες είναι ξένα κλειδιά.



Εικόνα 1 - Διάγραμμα βάσης δεομένων ηλεκτρονικού δισκοπωλείου

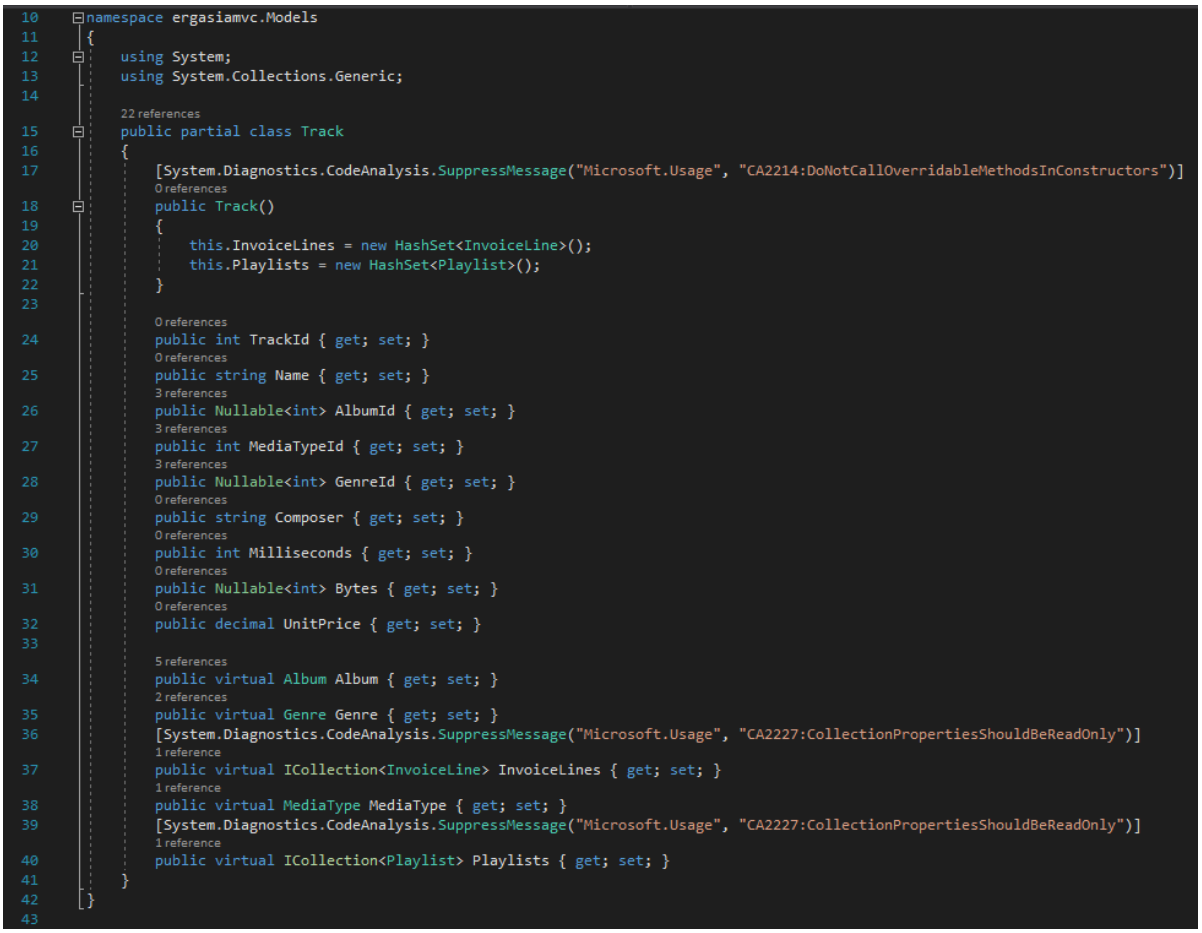
Οι πρώτοι 10 πίνακες αναπαραστούνται στο χώρο της εφαρμογής ως κλάσεις C# και τα αντικείμενά τους ως αντικείμενα των κλάσεων αυτών. Με άλλα λόγια, οι πρώτοι 10 πίνακες έχουν τα δικά τους Models μέσα στην εφαρμογή, ενώ τα αντικείμενα του πίνακα PlaylistTrack λειτουργούν ως properties στα αντικείμενα των πινάκων για τα οποία εκφράζουν σχέσεις. Στην Εικόνα 2 φαίνεται ένα τυπικό Model.



```
namespace ergasiamvc.Models
{
    using System;
    using System.Collections.Generic;

    11 references
    public partial class Invoice
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]
        0 references
        public Invoice()
        {
            this.InvoiceLines = new HashSet<InvoiceLine>();
        }

        0 references
        public int InvoiceId { get; set; }
        3 references
        public int CustomerId { get; set; }
        8 references
        public System.DateTime InvoiceDate { get; set; }
        0 references
        public string BillingAddress { get; set; }
        0 references
        public string BillingCity { get; set; }
        0 references
        public string BillingState { get; set; }
        0 references
        public string BillingCountry { get; set; }
        0 references
        public string BillingPostalCode { get; set; }
        0 references
        public decimal Total { get; set; }

        1 reference
        public virtual Customer Customer { get; set; }
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]
        1 reference
        public virtual ICollection<InvoiceLine> InvoiceLines { get; set; }
    }
}
```

Εικόνα 2 - Invoice model

Τα Models έχουν ένα property για κάθε μία από τις στήλες του πίνακα στη βάση στον οποίο απευθύνεται το κάθε ένα. Οι στήλες με απλά στοιχεία, όπως αριθμούς και strings, αντιπροσωπεύονται με στοιχειώδεις τύπους δεδομένων, όπως integers, floats και strings. Πιο σύνθετα δεδομένα αντιπροσωπεύονται με πιο σύνθετους τύπους. Για παράδειγμα, οι ημερομηνίες αντιπροσωπεύονται με την κλάση της C# DateTime, προσφέροντας έτσι την επεξεργασία τους με τις συναρτήσεις της κλάσης αυτής. Η στήλες με τα ξένα κλειδιά, που περιέχουν ids για αντικείμενα σε άλλους πίνακες, δεν αντιπροσωπεύονται ως απλοί integers, αλλά ως αντικείμενα της κατάλληλης κλάσης. Για παράδειγμα, στην Εικόνα 2 φαίνεται η κλάση Invoice, η οποία αντιπροσωπεύει τις αποδείξεις από τις αγορές των πελατών. Στη βάση δεδομένων, στον πίνακα Invoice υπάρχει η στήλη CustomerId, η οποία είναι foreign key που αναφέρεται στον πελάτη στον οποίο ανήκει η απόδειξη. Στη βάση αυτή πληροφορία αναπαρίσταται ως απλώς το id του πελάτη, ενώ στην εφαρμογή αυτή η πληροφορία αναπαρίσταται ως ένα αντικείμενο Customer, το οποίο περιέχει όλες τις πληροφορίες του πελάτη, δίνοντας έτσι στη εφαρμογή πρόσβαση σε έναν πελάτη μέσω μιας απόδειξής του. Αυτό γίνεται όταν ένα αντικείμενο "ανήκει" σε μόνο ένα άλλο. Στην περίπτωση που σε ένα αντικείμενο ανήκουν πολλά άλλα, αυτή η σχέση αντιπροσωπεύεται με μια λίστα αντικειμένων κατάλληλου τύπου, παρ' όλο που αυτό δεν φαίνεται στους πίνακες της βάσης. Πηγαίνοντας πάλι στην Εικόνα 2, βλέπουμε ότι η κλάση Invoice έχει ως property μια λίστα αντικειμένων τύπου InvoiceLine. Η κλάση InvoiceLine αντιπροσωπεύει τις καταχωρήσεις σε μια απόδειξη. Έχοντας αυτή τη λίστα ως property με αυτό τον τρόπο, η εφαρμογή μπορεί να έχει πρόσβαση σε όλες τις πληροφορίες των γραμμών μιας απόδειξης, μέσω της ίδιας της απόδειξης.



```
namespace ergasiamvc.Models
{
    using System;
    using System.Collections.Generic;

    22 references
    public partial class Track
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]
        0 references
        public Track()
        {
            this.InvoiceLines = new HashSet<InvoiceLine>();
            this.Playlists = new HashSet<Playlist>();
        }

        0 references
        public int TrackId { get; set; }
        0 references
        public string Name { get; set; }
        3 references
        public Nullable<int> AlbumId { get; set; }
        3 references
        public int MediaTypeId { get; set; }
        3 references
        public Nullable<int> GenreId { get; set; }
        0 references
        public string Composer { get; set; }
        0 references
        public int Milliseconds { get; set; }
        0 references
        public Nullable<int> Bytes { get; set; }
        0 references
        public decimal UnitPrice { get; set; }

        5 references
        public virtual Album Album { get; set; }
        2 references
        public virtual Genre Genre { get; set; }
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]
        1 reference
        public virtual ICollection<InvoiceLine> InvoiceLines { get; set; }
        1 reference
        public virtual MediaType MediaType { get; set; }
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]
        1 reference
        public virtual ICollection<Playlist> Playlists { get; set; }
    }
}
```

Εικόνα 3 - Track model

Στην Εικόνα 3 φαίνεται η κλάση Track. Η κλάση Track αποτελεί την αναπαράσταση του πίνακα Track της βάσης δεδομένων στην εφαρμογή, ο οποίος περιέχει τα μουσικά κομμάτια που πωλούνται στο ηλεκτρονικό δισκοπωλείο. Σε αυτή την κλάση βρίσκεται ένα παράδειγμα της μεταχείρισης σχέσεων πολλών προς πολλών στην εφαρμογή, δηλαδή πινάκων όπως ο PlaylistTrack. Ο πίνακας PlaylistTrack δείχνει ποια κομμάτια βρίσκονται σε ποιες playlists. Για αυτόν τον πίνακα δεν υπάρχει κάποιο Model, αλλά στο Model του πίνακα Track υπάρχει μια λίστα τύπου Playlist και στο μοντέλο του πίνακα Playlist υπάρχει μια λίστα τύπου Track. Έτσι η εφαρμογή μπορεί να προσπελάσει τα playlists στα οποία ανήκει ένα κομμάτι μέσω του κομματιού και τα κομμάτια ενός playlist μέσω του playlist.

3 Διαχείριση Δεδομένων

Η επεξεργασία των δεδομένων τη βάσης δεδομένων γίνεται μέσω ιστοσελίδων, Views, σε επίπεδο χρήστη, ενώ η λογική των ενεργειών πάνω στη βάση δεδομένων γίνεται μέσω ειδικών κλάσεων χειρισμού, των Controllers. Για κάθε πίνακα τη βάσης δεδομένων, εκτός από

τον πίνακα PlaylistTrack, υπάρχει μία βασική υποδομή που αναλαμβάνει την εκτέλεση διάφορων ενεργειών πάνω στα δεδομένα του. Για κάθε τύπο δεδομένων της βάσης, υπάρχουν πέντε Views, μία για την προβολή των δεδομένων ενός πίνακα σε μορφή λίστας, μία για την δημιουργία νέων αντικειμένων σε έναν πίνακα, μία για την προβολή των λεπτομερειών των αντικειμένων του πίνακα, μία για την επεξεργασία των λεπτομερειών αυτών και μία για τη διαγραφή αντικειμένων από έναν πίνακα. Αυτές οι Views λειτουργούν ως η γραφική διεπαφή του χρήστη με τη βάση δεδομένων. Οι επεμβάσεις στη βάση δεδομένων και η ανταπόκριση στο χρήστη μέσω αλλαγών στις Views πραγματοποιούνται από τους Controllers.

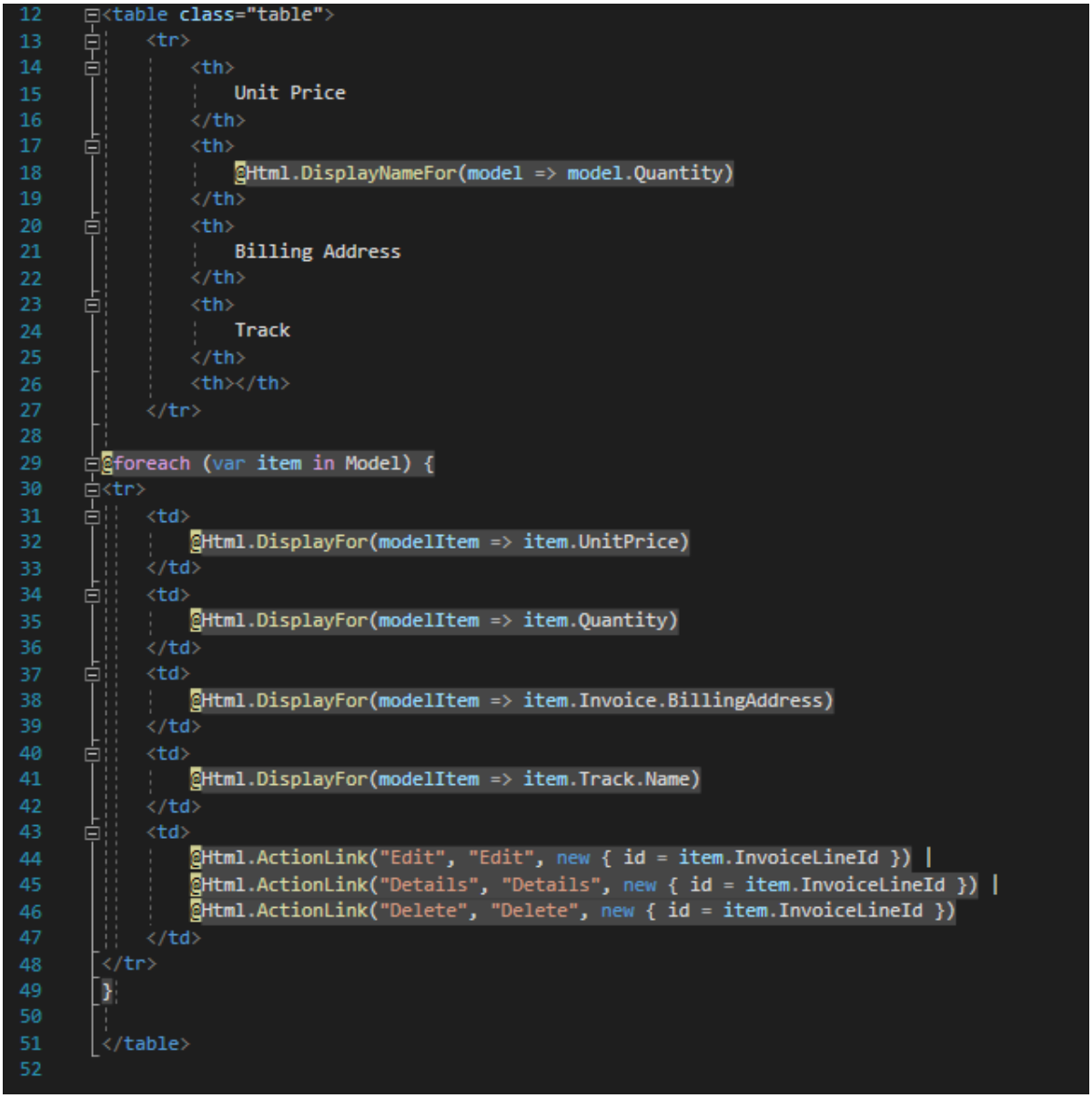
3.1 Προβολή δεδομένων σε λίστα

Για κάθε πίνακα της βάσης δεδομένων, εκτός του πίνακα PlaylistTrack, υπάρχει μία σελίδα όπου ο χρήστης μπορεί να δει τα δεδομένα του πίνακα σε μορφή λίστας. Στους controllers όλων των πινάκων, υπάρχει η συνάρτηση Index, η οποία φαίνεται στην Εικόνα 4. Η συνάρτηση Index τραβάει από τη βάση δεδομένων όλα τα αντικείμενα του πίνακα ενός τύπου δεδομένων. Στην Εικόνα 4 φαίνεται η συνάρτηση Index στον controller για το Model InvoiceLines. Η συνάρτηση τραβάει από τη βάση δεδομένων τις πληροφορίες για όλα τα αντικείμενα του πίνακα InvoiceLines, μαζί με τις πληροφορίες από τους πίνακες Invoice και Track που τους αντιστοιχούν.

```
18      public ActionResult Index()
19      {
20          var invoiceLines = db.InvoiceLines.Include(i => i.Invoice).Include(i => i.Track);
21          return View(invoiceLines.ToList());
22      }
```

Εικόνα 4 - Συνάρτηση Index στον controller του Model InvoiceLines

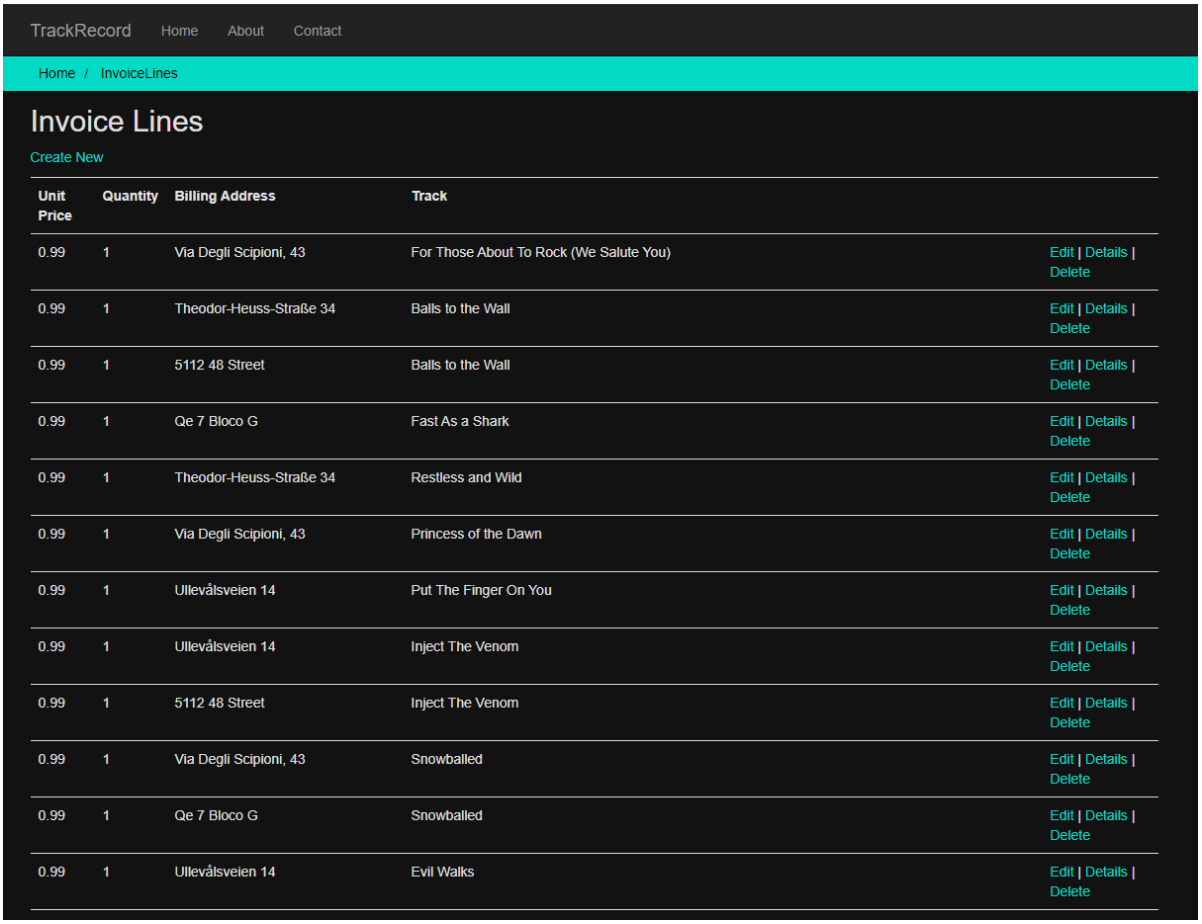
Έπειτα δημιουργείται η κατάλληλη View. Για κάθε Model υπάρχει μια View ονόματι Index, στην οποία παρουσιάζονται όλα τα αντικείμενα ενός πίνακα της βάσης δεδομένων σε μορφή λίστας. Σε αυτή τη λίστα, κάθε γραμμή, αντιπροσωπεύει ένα αντικείμενο στον πίνακα και κάθε στήλη μια στήλη του πίνακα της βάσης δεδομένων στον οποίο απευθύνεται η View. Στην τελευταία στήλη κάθε γραμμής της λίστας, υπάρχουν σύνδεσμοι για την εκτέλεση διαφόρων ενεργειών στο αντικείμενο που αντιπροσωπεύει η γραμμή. Στην Εικόνα 5 φαίνεται ο κώδικας που χτίζει μια τέτοια λίστα, στην προκειμένη περίπτωση για το Model InvoiceLine.



```
<table class="table">
    <tr>
        <th>
            Unit Price
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Quantity)
        </th>
        <th>
            Billing Address
        </th>
        <th>
            Track
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
<tr>
    <td>
        @Html.DisplayFor(modelItem => item.UnitPrice)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.Quantity)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.Invoice.BillingAddress)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.Track.Name)
    </td>
    <td>
        @Html.ActionLink("Edit", "Edit", new { id = item.InvoiceLineId }) |
        @Html.ActionLink("Details", "Details", new { id = item.InvoiceLineId }) |
        @Html.ActionLink("Delete", "Delete", new { id = item.InvoiceLineId })
    </td>
</tr>
}

</table>
```

Εικόνα 5 - Αυτός ο κώδικας χτίσει μια λίστα για την προβολή των αντικειμένων του πίνακα InvoiceLines

Το αποτέλεσμα του κώδικα της Εικόνας 5 φαίνεται στην Εικόνα 6.



Εικόνα 6 - Σελίδα προβολής αντικειμένων του πίνακα InvoiceLines

3.2 Δημιουργία νέων αντικειμένων

Για κάθε πίνακα της βάσης δεδομένων, εκτός του πίνακα PlaylistTrack, υπάρχει μία σελίδα όπου ο χρήστης μπορεί να προσθέσει νέα δεδομένα στον πίνακα. Στους controllers όλων των πινάκων, υπάρχει η συνάρτηση Create σε δύο μορφές. Η μία μορφή καλείται όταν ο χρήστης ζητάει να εισέλθει στη φόρμα δημιουργίας αντικειμένου και η άλλη καλείται όταν ο χρήστης καταχωρεί το νέο αντικείμενο. Στην Εικόνα 7 φαίνεται η πρώτη μορφή και αναφέρεται στο Model InvoiceLnes. Το αποτέλεσμα αυτής της συνάρτησης είναι να εμφανιστεί στο χρήστη μια φόρμα όπου μπορεί να συμπληρώσει τα στοιχεία του νέου αντικειμένου, η οποία φαίνεται στην Εικόνα 9. Μέσα στη συνάρτηση Create της Εικόνας 7, ετοιμάζονται κάποια ειδικά πεδία της φόρμας τα οποία θα χρησιμοποιηθούν κατά τη δημιουργία της κατάλληλης View. Στο παράδειγμα, ετοιμάζονται δύο drop-down menus, ένα για να μπορεί να επιλέξει ο χρήστης σε ποια απόδειξη απευθύνεται η νέα γραμμή που θέλει να φτιάξει από αυτές που υπάρχουν, και ένα για να μπορεί ο χρήστης να επιλέξει για ποιο κομμάτι από τα διαθέσιμα είναι αυτή η γραμμή.

```
40          public ActionResult Create()
41          {
42              ViewBag.InvoiceId = new SelectList(db.Invoices, "InvoiceId", "BillingAddress");
43              ViewBag.TrackId = new SelectList(db.Tracks, "TrackId", "Name");
44              return View();
45          }
```

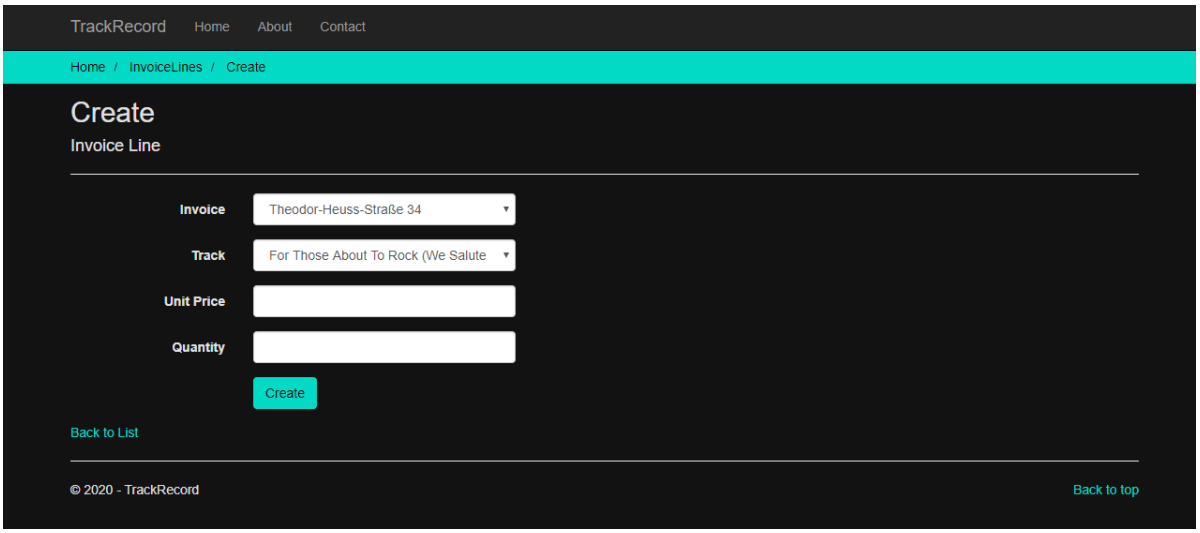
Εικόνα 7 - Συνάρτηση Create στον controller του Model InvoiceLines. Αυτή η μορφή της συνάρτησης καλείται όταν ο χρήστης εισέρχεται στη σελίδα δημιουργίας νέου αντικειμένου.

Στην Εικόνα 8 βλέπουμε τον κώδικα που χτίζει τη φόρμα δημιουργίας νέου αντικειμένου. Στις γραμμές 21 και 29 βλέπουμε το πως χρησιμοποιούνται τα πεδία που ετοιμάζει η συνάρτηση Create. Μία τέτοια View, ονόματι Create, υπάρχει για κάθε Model.

```
14  <div class="form-horizontal">
15      <h4>Invoice Line</h4>
16      <hr />
17      @Html.ValidationSummary(true, "", new { @class = "text-danger" })
18      <div class="form-group">
19          @Html.LabelFor(model => model.InvoiceId, "Invoice", htmlAttributes: new { @class = "control-label col-md-2" })
20          <div class="col-md-10">
21              @Html.DropDownList("InvoiceId", null, htmlAttributes: new { @class = "form-control" })
22              @Html.ValidationMessageFor(model => model.InvoiceId, "", new { @class = "text-danger" })
23          </div>
24      </div>
25
26      <div class="form-group">
27          @Html.LabelFor(model => model.TrackId, "Track", htmlAttributes: new { @class = "control-label col-md-2" })
28          <div class="col-md-10">
29              @Html.DropDownList("TrackId", null, htmlAttributes: new { @class = "form-control" })
30              @Html.ValidationMessageFor(model => model.TrackId, "", new { @class = "text-danger" })
31          </div>
32      </div>
33
34      <div class="form-group">
35          @Html.LabelFor(model => model.UnitPrice, "Unit Price", htmlAttributes: new { @class = "control-label col-md-2" })
36          <div class="col-md-10">
37              @Html.EditorFor(model => model.UnitPrice, new { htmlAttributes = new { @class = "form-control" } })
38              @Html.ValidationMessageFor(model => model.UnitPrice, "", new { @class = "text-danger" })
39          </div>
40      </div>
41
42      <div class="form-group">
43          @Html.LabelFor(model => model.Quantity, htmlAttributes: new { @class = "control-label col-md-2" })
44          <div class="col-md-10">
45              @Html.EditorFor(model => model.Quantity, new { htmlAttributes = new { @class = "form-control" } })
46              @Html.ValidationMessageFor(model => model.Quantity, "", new { @class = "text-danger" })
47          </div>
48      </div>
49
50      <div class="form-group">
51          <div class="col-md-offset-2 col-md-10">
52              <input type="submit" value="Create" class="btn btn-primary" />
53          </div>
54      </div>
55  </div>
56  }
```

Εικόνα 8 - Κώδικας που χτίζει τη φόρμα δημιουργίας νέου αντικειμένου για το Model InvoiceLine

Το αποτέλεσμα του κώδικα της Εικόνας 8 φαίνεται στην Εικόνα 9.



Εικόνα 9- Φόρμα δημιουργίας νέου InvoiceLine

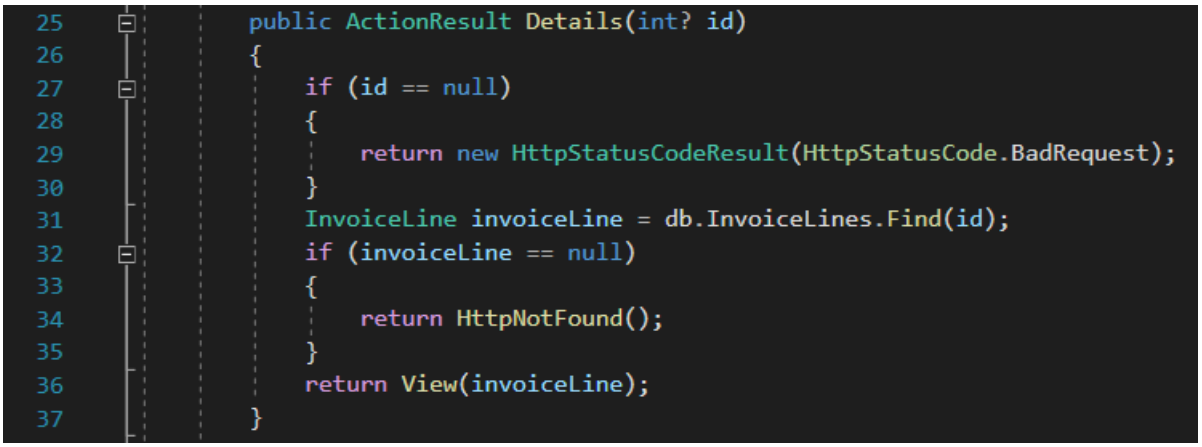
Όταν ο χρήστης πατήσει το κουμπί "Create", η μορφή της συνάρτησης Create που φαίνεται στην Εικόνα 10 καλείται. Η συνάρτηση ελέγχει για την εγκυρότητα των στοιχείων που καταχωρεί ο χρήστης, κι αν είναι τα αποθηκεύει στη βάση δεδομένων, στον κατάλληλο πίνακα, αλλιώς επιστρέφει το χρήστη πάλι στη φόρμα δημιουργίας και προβάλλει τα κατάλληλα μηνύματα λάθους.

```
50          [HttpPost]
51          [ValidateAntiForgeryToken]
            0 references
52          public ActionResult Create([Bind(Include = "InvoiceLineId,InvoiceId,TrackId,UnitPrice,Quantity")] InvoiceLine invoiceLine)
53          {
54              if (ModelState.IsValid)
55              {
56                  db.InvoiceLines.Add(invoiceLine);
57                  db.SaveChanges();
58                  return RedirectToAction("Index");
59              }
60
61              ViewBag.InvoiceId = new SelectList(db.Invoices, "InvoiceId", "BillingAddress", invoiceLine.InvoiceId);
62              ViewBag.TrackId = new SelectList(db.Tracks, "TrackId", "Name", invoiceLine.TrackId);
63              return View(invoiceLine);
64          }
```

Εικόνα 10 - Η μορφή της συνάρτησης Create που καλείται όταν ο χρήστης καταχωρεί ένα αντικείμενο

3.3 Προβολή λεπτομερειών αντικειμένου

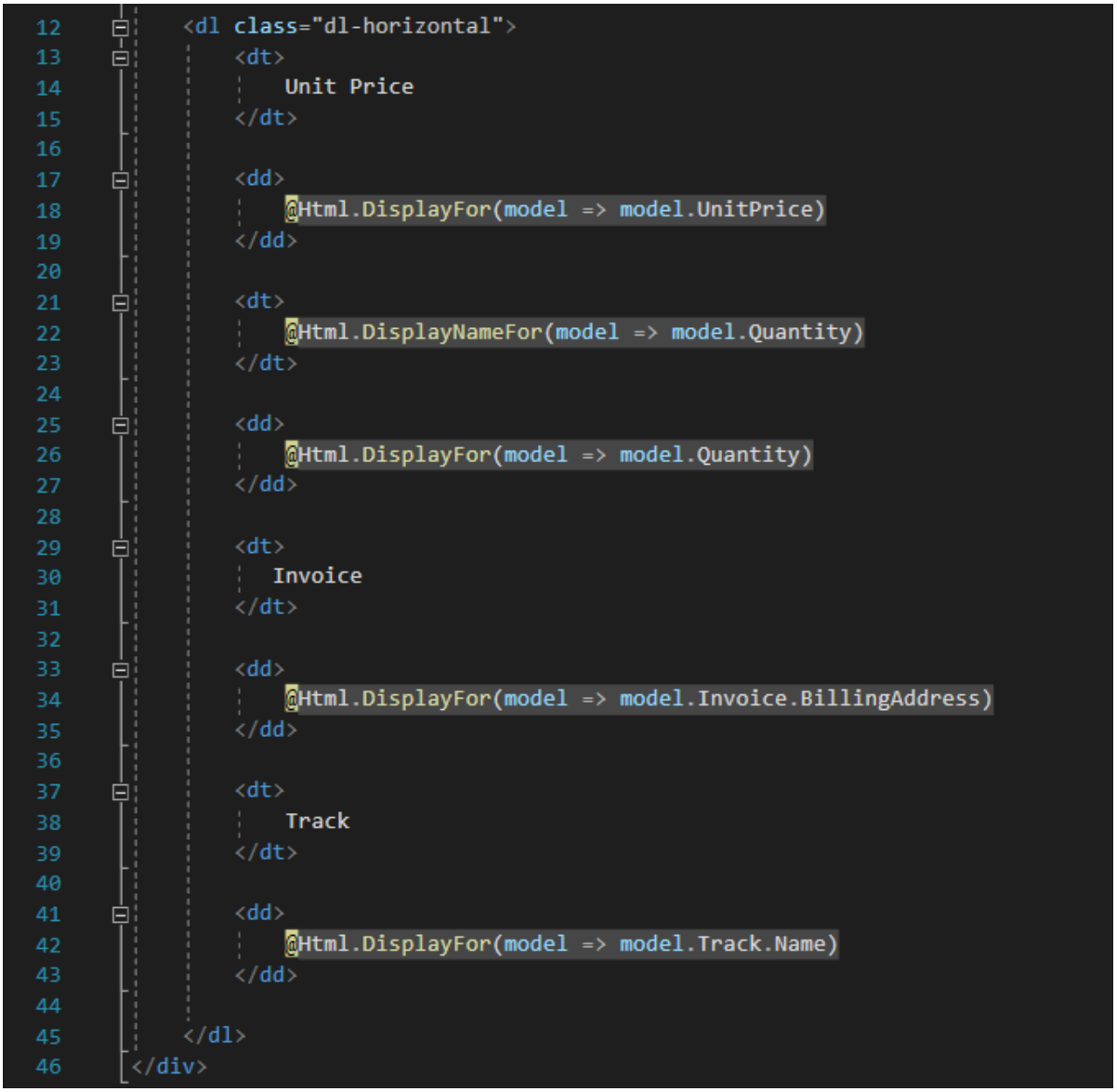
Για κάθε πίνακα της βάσης δεδομένων, εκτός του πίνακα PlaylistTrack, υπάρχει μία σελίδα όπου ο χρήστης μπορεί να δει τις λεπτομέρειες ενός αντικειμένου του πίνακα. Στους controllers όλων των πινάκων, υπάρχει η συνάρτηση Details. Η συνάρτηση Details για το Model InvoiceLine φαίνεται στην Εικόνα 11. Η συνάρτηση Details τραβά από τη βάση δεδομένων τις πληροφορίες του αντικειμένου που θέλει να δει ο χρήστης.



```
25      public ActionResult Details(int? id)
26      {
27          if (id == null)
28          {
29              return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
30          }
31          InvoiceLine invoiceLine = db.InvoiceLines.Find(id);
32          if (invoiceLine == null)
33          {
34              return HttpNotFound();
35          }
36          return View(invoiceLine);
37      }
```

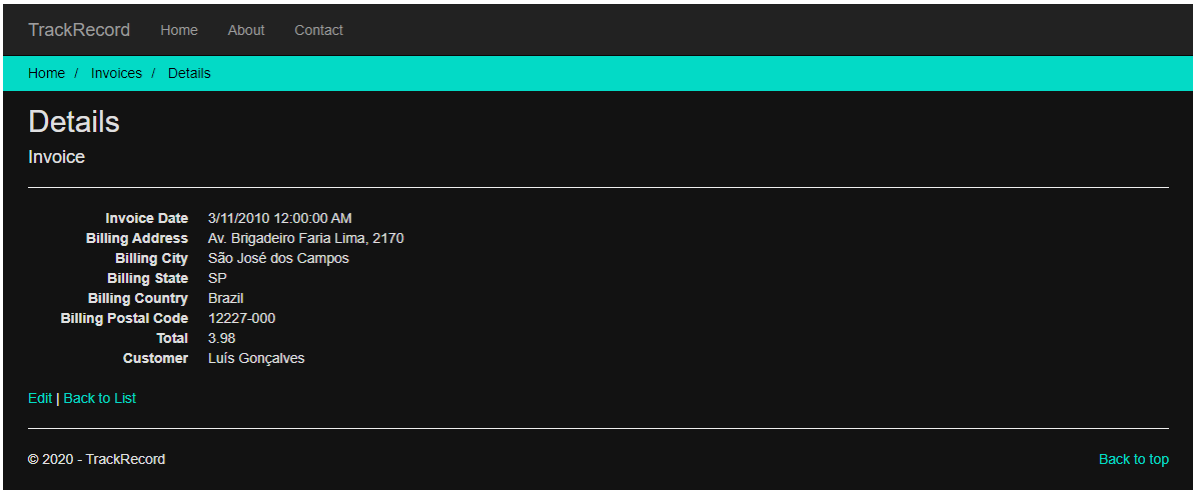
Εικόνα 11 - Συνάρτηση Details στον controller του Model InvoiceLine

Στην Εικόνα 12 φαίνεται ο κώδικας που χτίζει μια σελίδα προβολής λεπτομερειών και στην Εικόνα 13 φαίνεται το αποτέλεσμα αυτού του κώδικα.



```
<dl class="dl-horizontal">
    <dt>
        Unit Price
    </dt>

    <dd>
        @Html.DisplayFor(model => model.UnitPrice)
    </dd>

    <dt>
        @Html.DisplayNameFor(model => model.Quantity)
    </dt>

    <dd>
        @Html.DisplayFor(model => model.Quantity)
    </dd>

    <dt>
      Invoice
    </dt>

    <dd>
        @Html.DisplayFor(model => model.Invoice.BillingAddress)
    </dd>

    <dt>
        Track
    </dt>

    <dd>
        @Html.DisplayFor(model => model.Track.Name)
    </dd>

</dl>
</div>
```

Εικόνα 12 - Details View για το Model InvoiceLine



Εικόνα 13 - Σελίδα προβολής λεπτομερειών για το Model InvoiceLine

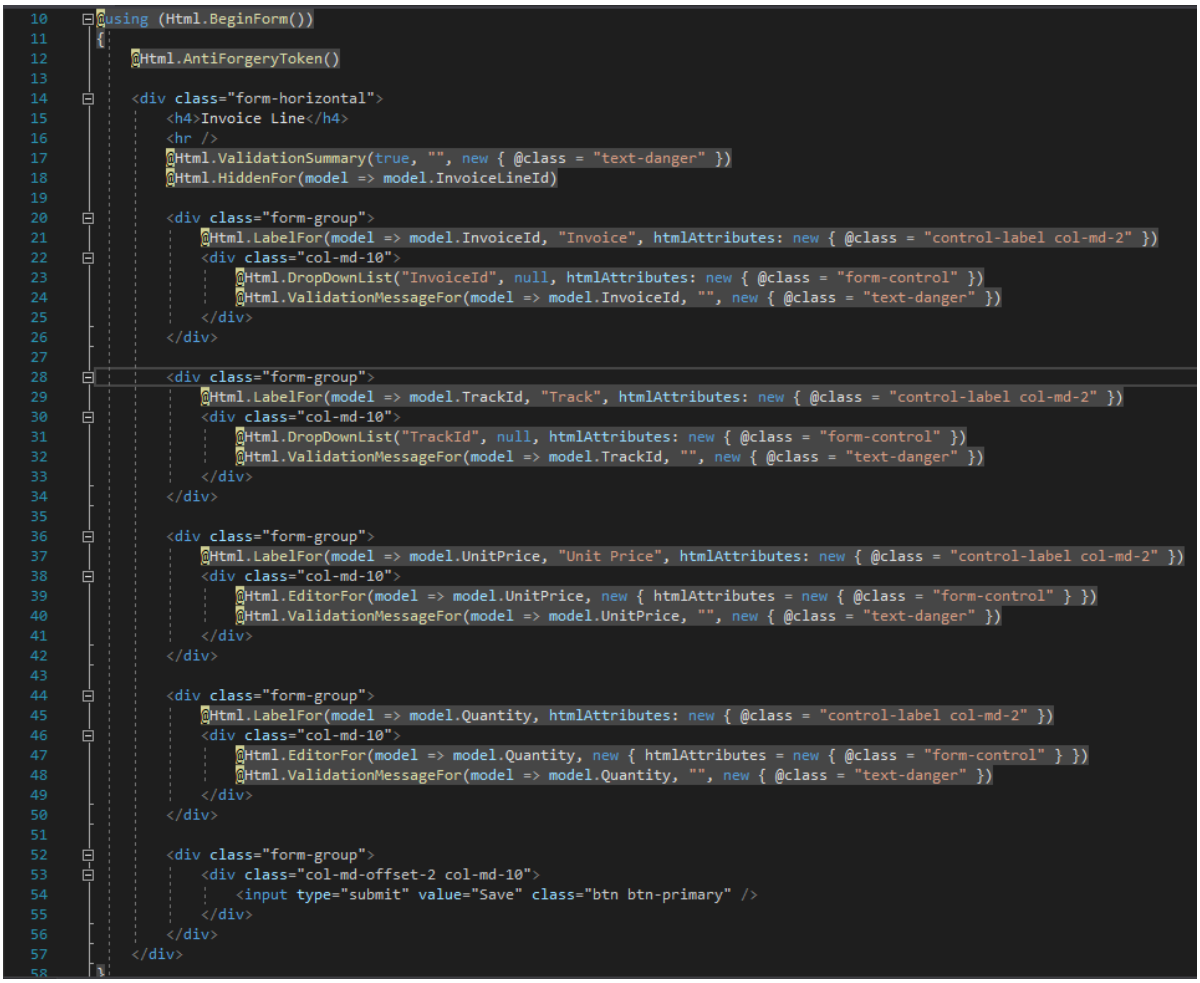
3.4 Τροποποίηση λεπτομερειών αντικειμένου

Για κάθε πίνακα της βάσης δεδομένων, εκτός του πίνακα PlaylistTrack, υπάρχει μία σελίδα όπου ο χρήστης μπορεί να τροποποιήσει τις λεπτομέρειες ενός αντικειμένου του πίνακα. Στους controllers όλων των πινάκων, υπάρχει η συνάρτηση Edit. Η συνάρτηση Edit έχει δύο μορφές, η πρώτη μορφή καλείται όταν ο χρήστης επιλέγει ένα αντικείμενο για τροποποίηση και η δεύτερη όταν ο χρήστης αποθηκεύει τις αλλαγές σε ένα αντικείμενο. Η πρώτη μορφή φαίνεται στην Εικόνα 14 και σε αυτή την περίπτωση αναφέρεται στο Model InvoiceLine. Η συνάρτηση ανακτά της πληροφορίες του αντικειμένου που θέλει να τροποποιήσει ο χρήστης από βάση δεδομένων και ετοιμάζει πληροφορίες για κάποια ειδικά πεδία. Σε αυτή την περίπτωση ετοιμάζει τις επιλογές για κάποια drop-down menus.

```
67      public ActionResult Edit(int? id)
68      {
69          if (id == null)
70          {
71              return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
72          }
73          InvoiceLine invoiceLine = db.InvoiceLines.Find(id);
74          if (invoiceLine == null)
75          {
76              return HttpNotFound();
77          }
78          ViewBag.InvoiceId = new SelectList(db.Invoices, "InvoiceId", "BillingAddress", invoiceLine.InvoiceId);
79          ViewBag.TrackId = new SelectList(db.Tracks, "TrackId", "Name", invoiceLine.TrackId);
80          return View(invoiceLine);
81      }
```

Εικόνα 14 - Μορφή της συνάρτησης Edit για το Model InvoiceLine η οποία καλείται όταν ο χρήστης επιλέγει ένα αντικείμενο για τροποποίηση.

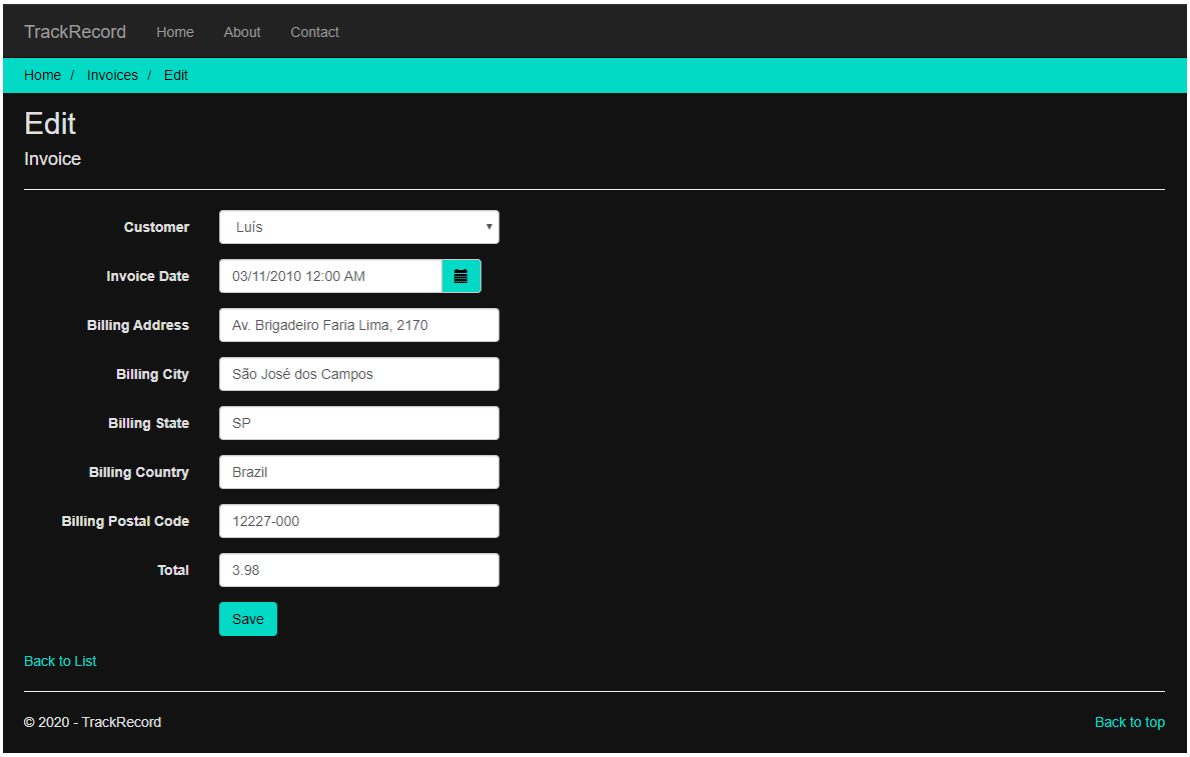
Στην Εικόνα 15 φαίνεται ο κώδικας που ετοιμάζει μια φόρμα τροποποίησης. Είναι παρόμοιος με τον κώδικα που δημιουργεί τη φόρμα δημιουργίας νέων αντικειμένων, αλλά γεμίζει τα πεδία της φόρμας με τις πληροφορίες του αντικειμένου που θέλει να τροποποιήσει ο χρήστης.



```
@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Invoice Line</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.InvoiceLineId)

        <div class="form-group">
            @Html.LabelFor(model => model.InvoiceId, "Invoice", htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("InvoiceId", null, htmlAttributes: new { @class = "form-control" })
                @Html.ValidationMessageFor(model => model.InvoiceId, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.TrackId, "Track", htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("TrackId", null, htmlAttributes: new { @class = "form-control" })
                @Html.ValidationMessageFor(model => model.TrackId, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.UnitPrice, "Unit Price", htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.UnitPrice, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.UnitPrice, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Quantity, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Quantity, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Quantity, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-primary" />
            </div>
        </div>
    </div>
```

Εικόνα 15 - Κώδικας που δημιουργεί τη φόρμα τροποποίησης για αντικείμενα του Model InvoiceLine

Στην Εικόνα 16 φαίνεται το αποτέλεσμα του παραπάνω κώδικα.



Εικόνα 16 - Σελίδα τροποποίησης για το Model InvoiceLine

Όταν ο χρήστης πατήσει το κουμπί "Save" για να αποθηκεύσει της αλλαγές, καλείτε η δεύτερη μορφή της συνάρτησης Edit, η οποία φαίνεται στην Εικόνα 17 για το Model InvoiceLine. Η συνάρτηση αυτή ελέγχει για την εγκυρότητα των πληροφοριών που καταχωρεί ο χρήστης, τις καταχωρεί αν είναι έγκυρες ή επιστρέφει το χρήστη πίσω στη φόρμα αν δεν είναι.

```
86          [HttpPost]
87          [ValidateAntiForgeryToken]
            0 references
88          public ActionResult Edit([Bind(Include = "InvoiceLineId,InvoiceId,TrackId,UnitPrice,Quantity")] InvoiceLine invoiceLine)
89          {
90              if (ModelState.IsValid)
91              {
92                  db.Entry(invoiceLine).State = EntityState.Modified;
93                  db.SaveChanges();
94                  return RedirectToAction("Index");
95              }
96              ViewBag.InvoiceId = new SelectList(db.Invoices, "InvoiceId", "BillingAddress", invoiceLine.InvoiceId);
97              ViewBag.TrackId = new SelectList(db.Tracks, "TrackId", "Name", invoiceLine.TrackId);
98              return View(invoiceLine);
99          }
```

Εικόνα 17 - Αυτή η συνάρτηση καλείται όταν ο χρήστης προσπαθεί να αποθηκεύσει τις αλλαγές που έχει κάνει σε ένα αντικείμενο

3.5 Διαγραφή αντικειμένου

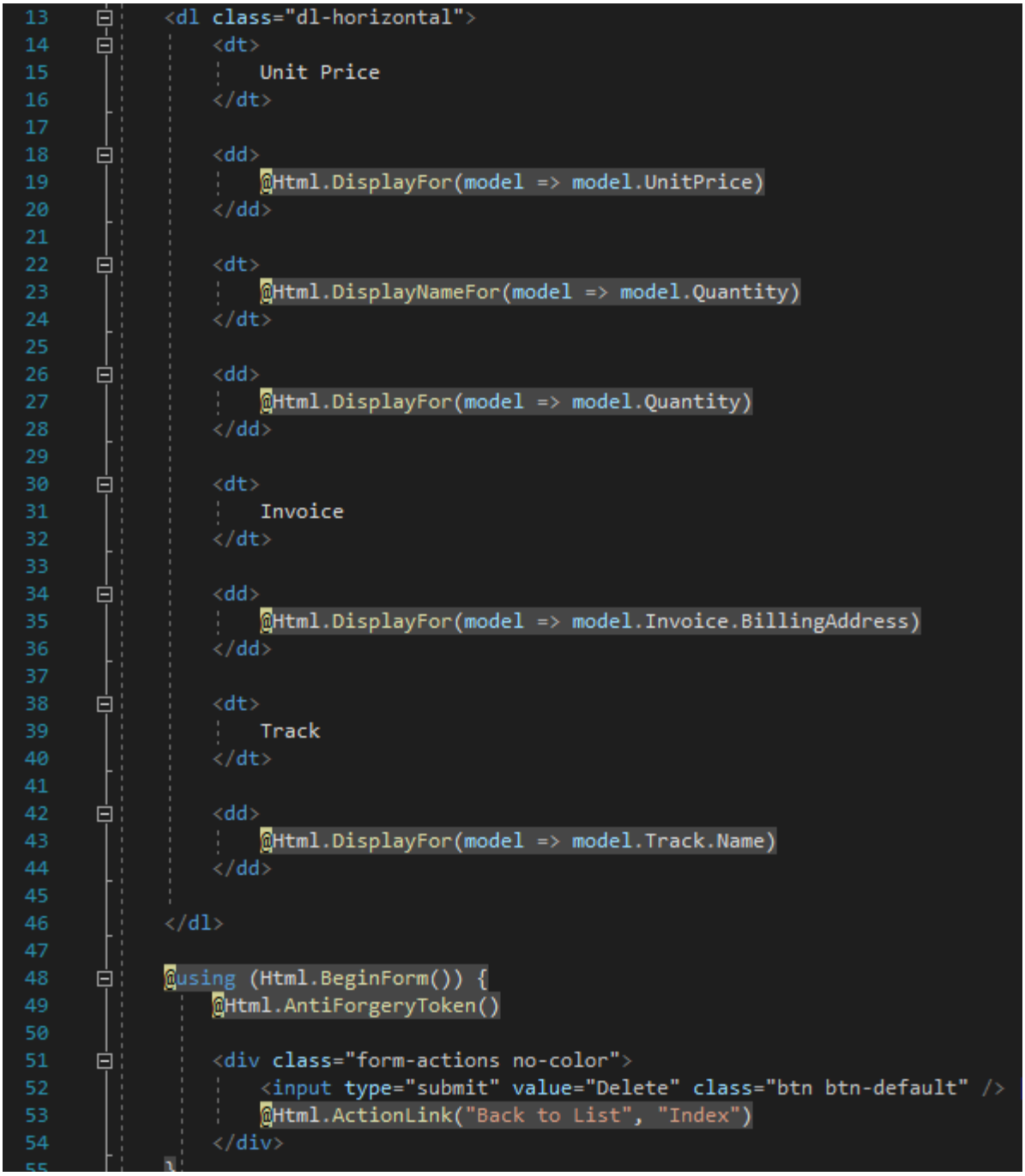
Για κάθε πίνακα της βάσης δεδομένων, εκτός του πίνακα PlaylistTrack, υπάρχει μία σελίδα όπου ο χρήστης μπορεί διαγράψει ένα αντικείμενο του πίνακα. Στους controllers όλων των πινάκων, υπάρχουν δύο συναρτήσεις, οι οποίες καθιστούν αυτή τη λειτουργικότητα δυνατή, η Delete και η DeleteConfirmed. Η συνάρτηση Delete ανακτά από τη βάση

δεδομένων τις πληροφορίες του αντικειμένου που θέλει να διαγράψει ο χρήστης και τις προβάλλει σε μία σελίδα, μέσω της οποίας ο χρήστης μπορεί να διαγράψει το αντικείμενο. Η συνάρτηση Delete για το Model InvoiceLine φαίνεται στην Εικόνα 18.

```
102     public ActionResult Delete(int? id)
103     {
104         if (id == null)
105         {
106             return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
107         }
108         InvoiceLine invoiceLine = db.InvoiceLines.Find(id);
109         if (invoiceLine == null)
110         {
111             return HttpNotFound();
112         }
113         return View(invoiceLine);
114     }
```

Εικόνα 18 - Συνάρτηση Delete για το Model InvoiceLine

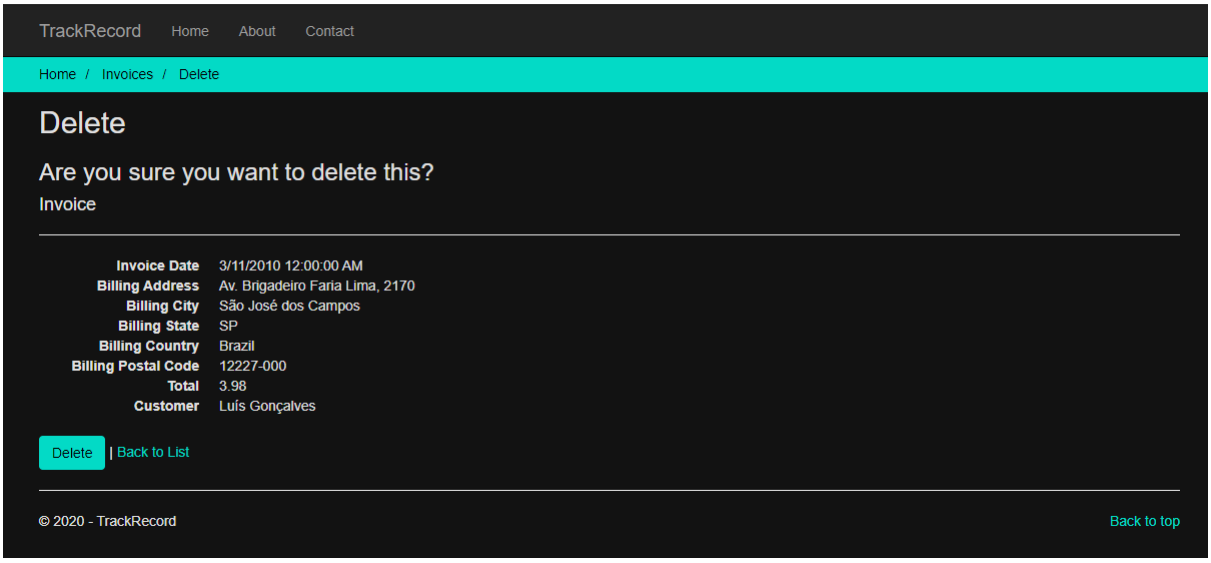
Στην Εικόνα 19 φαίνεται ο κώδικας που χτίζει της σελίδα διαγραφής. Είναι παρόμοιος με τον κώδικα που χτίζει τη σελίδα προβολής λεπτομερειών.



```
<dl class="dl-horizontal">
    <dt>
        Unit Price
    </dt>

    <dd>
        @Html.DisplayFor(model => model.UnitPrice)
    </dd>

    <dt>
        @Html.DisplayNameFor(model => model.Quantity)
    </dt>

    <dd>
        @Html.DisplayFor(model => model.Quantity)
    </dd>

    <dt>
        Invoice
    </dt>

    <dd>
        @Html.DisplayFor(model => model.Invoice.BillingAddress)
    </dd>

    <dt>
        Track
    </dt>

    <dd>
        @Html.DisplayFor(model => model.Track.Name)
    </dd>

</dl>

@using (Html.BeginForm()) {
    @Html.AntiForgeryToken()

    <div class="form-actions no-color">
        <input type="submit" value="Delete" class="btn btn-default" />
        @Html.ActionLink("Back to List", "Index")
    </div>
}
```

Εικόνα 19 - Κώδικας που χτίζει τη σελίδα διαγραφής για αντικείμενα του Model InvoiceLine

Στην Εικόνα 20 φαίνεται το αποτέλεσμα του παραπάνω κώδικα.



Εικόνα 20 - Σελίδα διαγραφής για αντικείμενα του Model InvoiceLine

Όταν ο χρήστης πατήσει το κουμπί "Delete" για να διαγράψει ένα αντικείμενο, καλείται η συνάρτηση DeleteConfirmed, ο κώδικας της οποίας φαίνεται στην Εικόνα 21. Η συνάρτηση βρίσκει το αντικείμενο που θέλει να διαγράψει ο χρήστης στη βάση δεδομένων και το διαγράφει, έπειτα επιστρέφει το χρήστη στη λίστα με τα αντικείμενα.

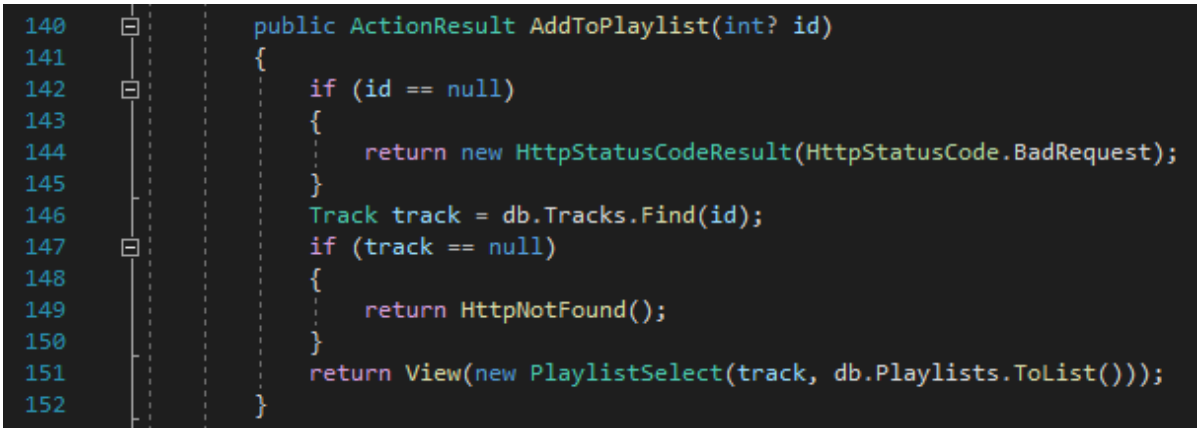
```
117         [HttpPost, ActionName("Delete")]
118         [ValidateAntiForgeryToken]
            0 references
119         public ActionResult DeleteConfirmed(int id)
120         {
121             InvoiceLine invoiceLine = db.InvoiceLines.Find(id);
122             db.InvoiceLines.Remove(invoiceLine);
123             db.SaveChanges();
124             return RedirectToAction("Index");
125         }
```

Εικόνα 21 - Συνάρτηση DeleteConfirmed για το Model InvoiceLine

3.6 Διαχείριση playlist

Όπως έχει προειπωθεί, η παραπάνω διαδικασίες διαχείρισης δεδομένων απευθύνονται σε όλους τους πίνακες της βάσης δεδομένων, εκτός από τον πίνακα PlaylistTrack. Ο πίνακας PlaylistTrack, δείχνει ποια κομμάτια βρίσκονται σε ποιες playlists. Η προσθαφαίρεση κομματιών από τις playlists γίνεται ειδικές συναρτήσεις και Views για αυτό το σκοπό, μέσα από τους controllers των Models Track και Playlist.

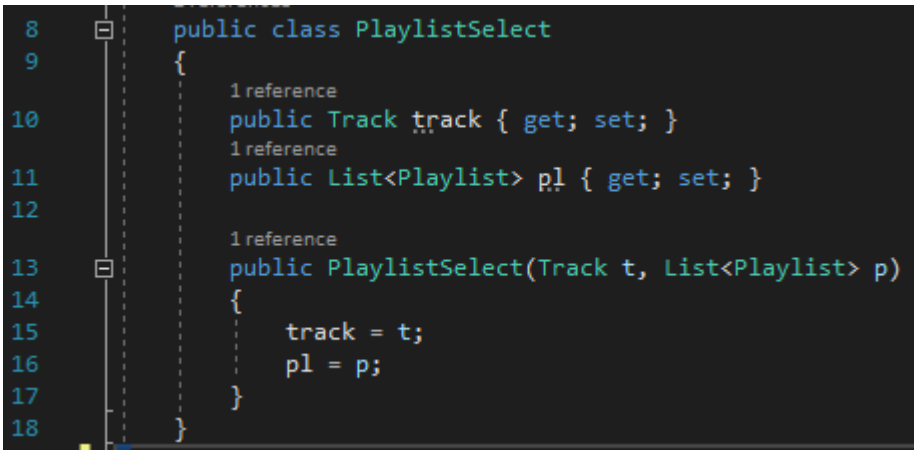
Η προσθήκη κομματιών σε playlist γίνεται μέσω των συναρτήσεων AddToPlaylist και AddToPlaylistConfirmed, οι οποίες βρίσκονται στον controller του Model Track. Η συνάρτηση AddToPlaylist προβάλλει στο χρήστη μια φόρμα, μέσω της οποίας μπορεί να επιλέξει σε πιο playlist θέλει να προσθέσει ένα κομμάτι. Ο κώδικας της συνάρτησης φαίνεται στην Εικόνα 22.



```
140         public ActionResult AddToPlaylist(int? id)
141         {
142             if (id == null)
143             {
144                 return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
145             }
146             Track track = db.Tracks.Find(id);
147             if (track == null)
148             {
149                 return HttpNotFound();
150             }
151             return View(new PlaylistSelect(track, db.Playlists.ToList()));
152         }
```

Εικόνα 22 - Συνάρτηση AddToPlaylist

Η συνάρτηση στέλνει στη View που χτίζει τη σελίδα επιλογής playlist τις πληροφορίες του κομματιού που θέλει να προσθέσει ο χρήστης σε ένα playlist και όλα τα playlist που υπάρχουν στη βάση, πακεταρισμένα σε ένα Model που χρησιμοποιείται μόνο για αυτή τη διαδικασία, το PlaylistSelect. Το Model έχει δύο properties, ένα αντικείμενο τύπου Track και μία λίστα από αντικείμενα τύπου Playlist. Το Model φαίνεται στην Εικόνα 23.



```
8   public class PlaylistSelect
9   {
        1 reference
10      public Track track { get; set; }
        1 reference
11      public List<Playlist> pl { get; set; }
12
        1 reference
13      public PlaylistSelect(Track t, List<Playlist> p)
14      {
15          track = t;
16          pl = p;
17      }
18  }
```

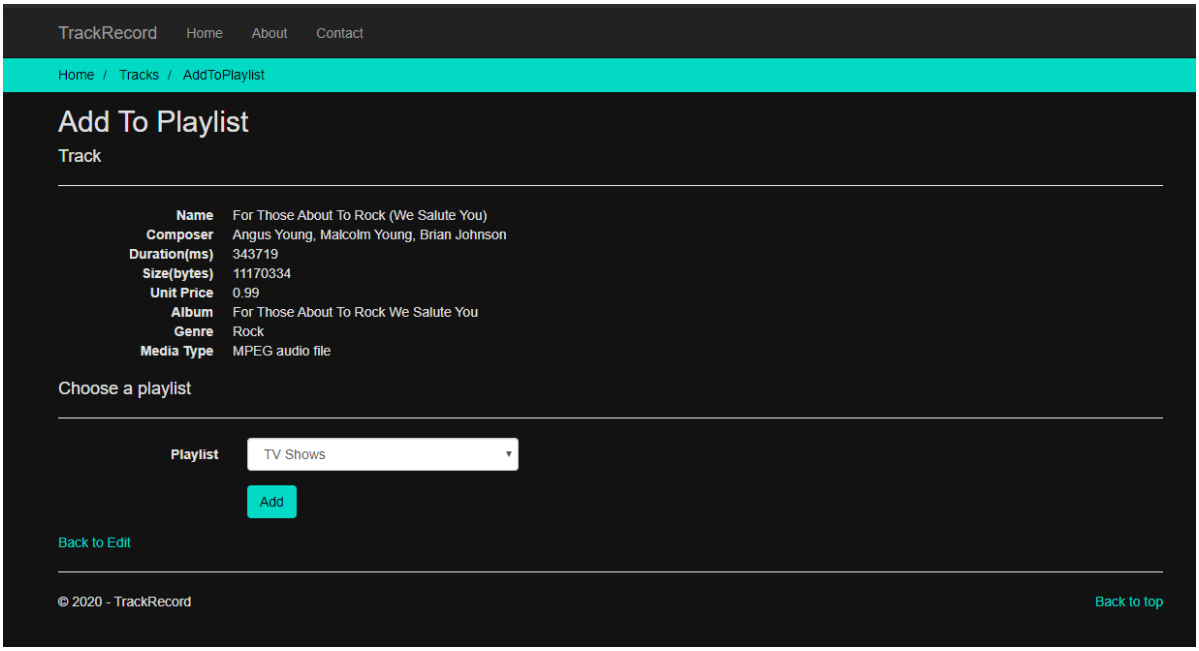
Εικόνα 23 - PlaylistSelect Model

Η φόρμα επιλογής playlist χτίζεται από τον κώδικα που φαίνεται στην Εικόνα 24. Ο κώδικας δημιουργεί ένα drop-down menu από το οποίο ο χρήστης μπορεί να επιλέξει σε ποιο playlist θέλει να καταχωρήσει το τρέχον κομμάτι. Η επιλογές φιλτράρονται, κάνοντας προς επιλογή διαθέσιμα μόνο τα playlists στα οποία δε βρίσκεται το τρέχον κομμάτι.

```
@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Choose a playlist</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.track.TrackId)

        <div class="form-group">
            <label class="control-label col-md-2" for="test">Playlist</label>
            <div class="col-md-10">
                <select class="form-control" id="test" name="playlistid">
                    @foreach (var item in Model.pl)
                    {
                        if (!Model.track.Playlists.Contains(item)){
                            <option value="@item.PlaylistId">@item.Name</option>
                        }
                    }
                </select>
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Add" class="btn btn-primary" />
            </div>
        </div>
    </div>
}
```

Εικόνα 24 - Κώδικας που δημιουργεί το drop-down menu επιλογής playlist

Το αποτέλεσμα του παραπάνω κώδικα φαίνεται στην Εικόνα 25.



Εικόνα 25 - Σελίδα προσθήκης κομματιού σε playlist

Όταν ο χρήστης πατήσει το κουμπί "Add", για να προσθέσει το κομμάτι στο playlist που έχει επιλέξει, καλείται η συνάρτηση AddToPlaylistConfirmed, η οποία φαίνεται στην Εικόνα 26. Η συνάρτηση προσθέτει το κομμάτι στο playlist που επέλεξε ο χρήστης. Στη ουσία, δημιουργείται μια καινούρια είσοδος στον πίνακα PlaylistTrack, με το id του κομματιού και το id του playlist που επιλέχθηκε. Οι αλλαγές αποθηκεύονται και ο χρήστης επιστρέφει στη φόρμα επιλογής playlist, ώστε να προσθέσει το κομμάτι σε κάποιο άλλο playlist αν θέλει.

```
154         [HttpPost, ActionName("AddToPlaylist")]
155         [ValidateAntiForgeryToken]
            0 references
156         public ActionResult AddConfirmed(int id, int playlistid)
157         {
158             Track track = db.Tracks.Find(id);
159             db.Playlists.Find(playlistid).Tracks.Add(db.Tracks.Find(id));
160             db.SaveChanges();
161             return RedirectToAction("AddToPlaylist/" + id);
162         }
```

Εικόνα 26 - Συνάρτηση AddToPlaylistConfirmed

Η αφαίρεση κομματιών από ένα playlist γίνεται μέσω της συνάρτησης RemoveFromPlaylist στον controller του Model Playlist. Η συνάρτηση φαίνεται στην Εικόνα 27. Αφαιρεί το κομμάτι από το playlist και επιστρέφει το χρήστη στη σελίδα τροποποίησης του playlist.

```
118     public ActionResult RemoveFromPlaylist(int playlistid, int trackid)
119     {
120         db.Playlists.Find(playlistid).Tracks.Remove(db.Tracks.Find(trackid));
121         db.SaveChanges();
122         return RedirectToAction("Edit\\"+playlistid);
123     }
```

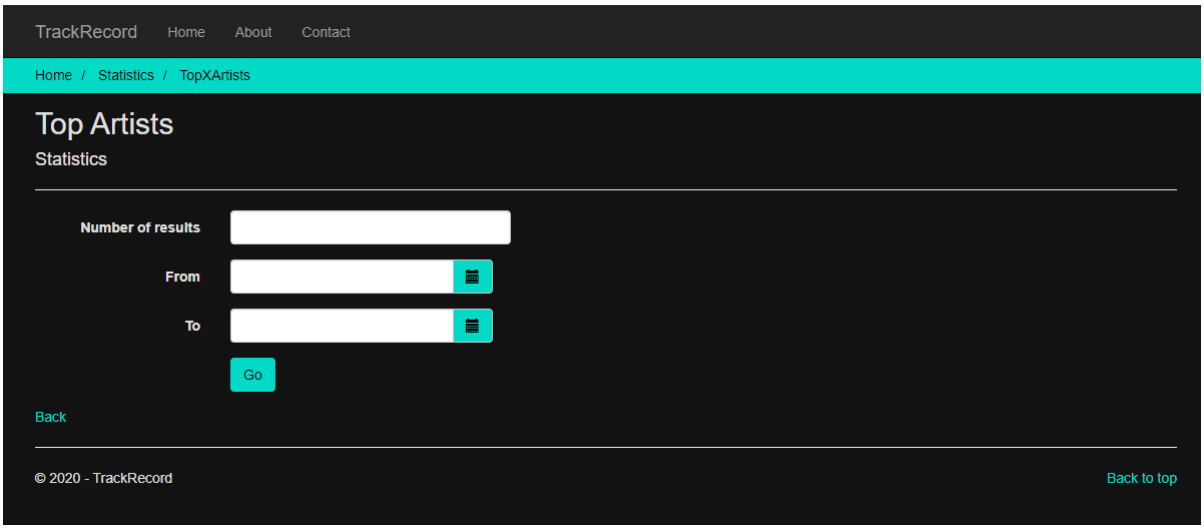
Εικόνα 27 - Συνάρτηση RemoveFromPlaylist

4 Εξαγωγή Στατισιτικών

Η εφαρμογή προσφέρει τη δυνατότητα εξαγωγής κάποιων στατιστικών από τη βάση δεδομένων. Ο χρήστης μπορεί να δει τρεις τύπους στατιστικών. Μπορεί να δει τους καλλιτέχνες που οι οι δίσκοι τους είναι ανάμεσα στους x πρώτους σε πωλήσεις, όπου το x το ορίζει ο χρήστης. Μπορεί να δει τα 10 κορυφαία σε πωλήσεις κομμάτια. Μπορεί να δει τα κορυφαία είδη στις προτιμήσεις των πελατών. Η όλη διαδικασία υποστηρίζεται από έναν controller, τον StatisticsController, ο οποίος αναλαμβάνει την εκτέλεση διάφορων queries πάνω στη βάση δεδομένων, Models, τα οποία χρησιμοποιούνται για την αναπαράσταση των δεδομένων των στατιστικών, και Views μέσω των οποίων ο χρήστης ζητά και βλέπει τα στατιστικά.

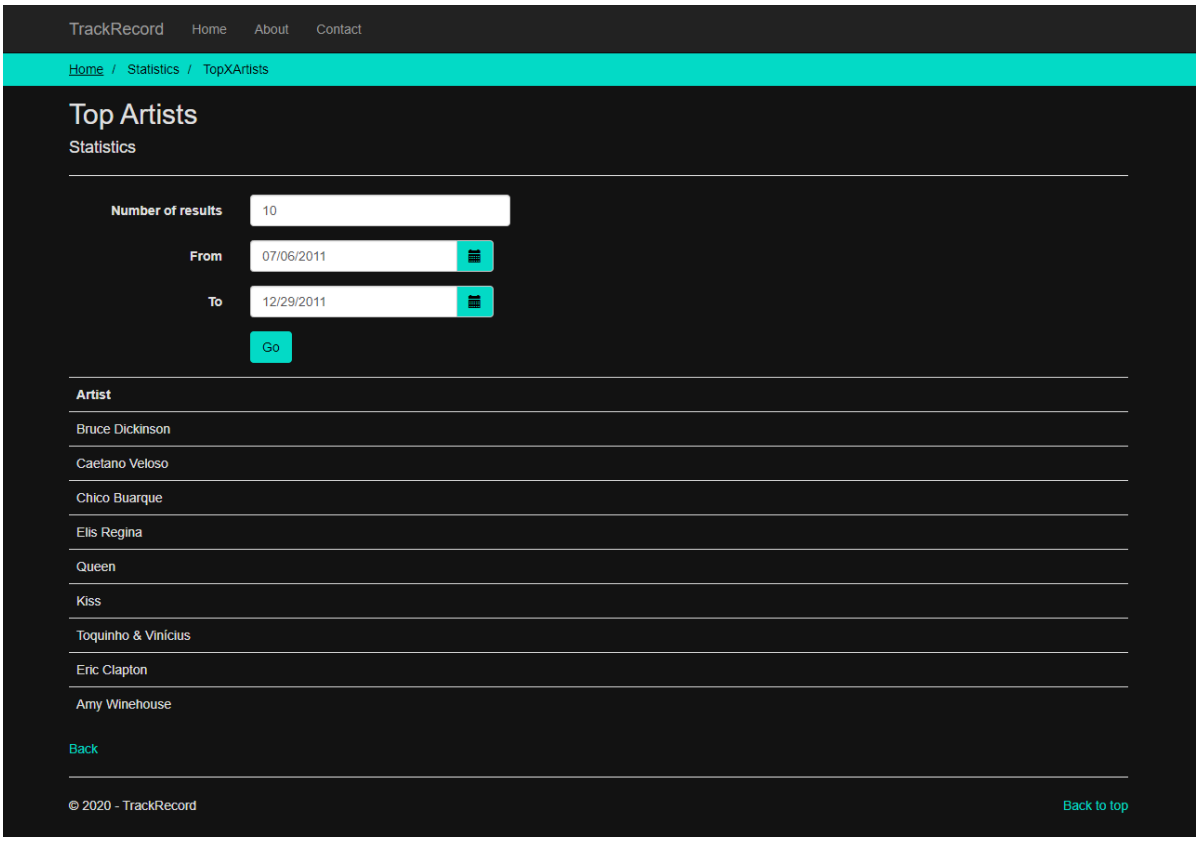
4.1 Κορυφαίοι καλλιτέχνες

Ένα από τα στατιστικά που μπορεί να δει ο χρήστης, είναι οι καλλιτέχνες που οι δίσκοι τους βρίσκονται στους κορυφαίους x σε πωλήσεις. Τον αριθμό x τον ορίζει ο χρήστης και είναι ο αριθμός των δίσκων οι οποίοι θα ληφθούν υπ' όψη. Επίσης, ο χρήστης μπορεί να επιλέξει το χρονικό διάστημα για το οποίο θέλει να ανακτήσει αποτελέσματα. Στην Εικόνα 28 φαίνεται η φόρμα που συμπληρώνει ο χρήστης για να παραμετροποιήσει την εξαγωγή στατιστικών.



Εικόνα 28 - Φόρμα ανάκτησης κορυφαίων καλλιτεχνών

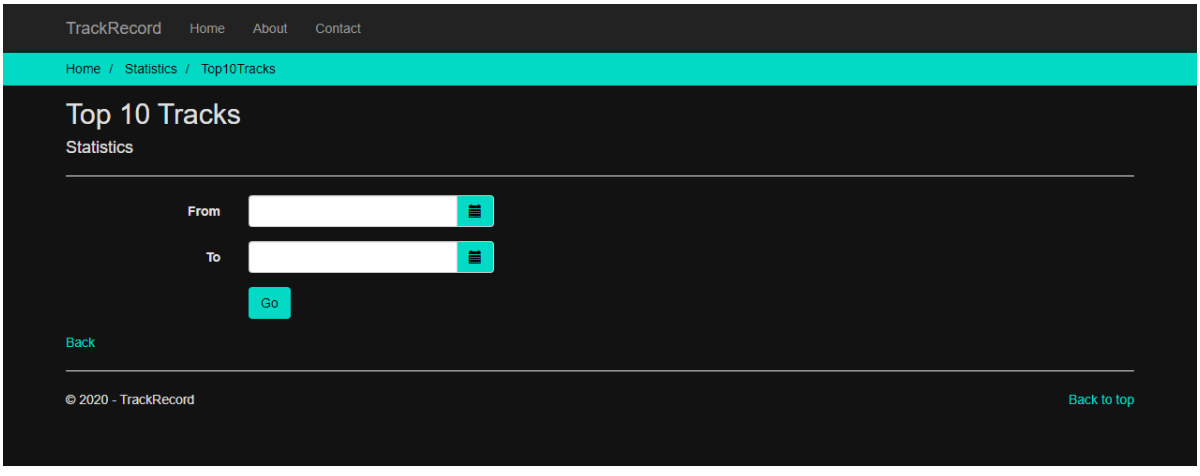
Ο χρήστης μπορεί να αφήσει κενό όποιο πεδίο θέλει. Τα κενά πεδία δεν λαμβάνονται υπ' όψη. Όταν πατήσει το κουμπί "Go" καλείται η συνάρτηση GetArtists, η οποία βρίσκεται στον StatisticsController. Η συνάρτηση βρίσκει από τον πίνακα InvoiceLine πόσες φορές έχει πουληθεί ένα κομμάτι και μετά ομαδοποιεί τα κομμάτια από τους ίδιους δίσκους για να βρει πόσα κομμάτια από κάθε δίσκο έχουν πουληθεί. Λαμβάνει υπ' όψη μόνο τα αντικείμενα του πίνακα InvoiceLine που ανήκουν σε αποδείξεις που έχουν καταχωρηθεί μέσα στα χρονικά πλαίσια που έχει ορίσει ο χρήστης. Ταξινομεί τους δίσκους με βάση τις πωλήσεις και για x από αυτούς παίρνει τους μοναδικούς καλλιτέχνες. Τα αποτελέσματα της συνάρτησης φαίνονται στην Εικόνα 29. Οι καλλιτέχνες εμφανίζονται στο χρήστη σε μορφή λίστας, κάτω από τη φόρμα.



Εικόνα 29 - Αποτελέσματα ανάκτησης κορυφαίων καλιτεχνών που οι δίσκοι τους βρίσκονται ανάμεσα στους 10 κορυφαίους για την περίοδο 07/06/2011 - 12/29/2011.

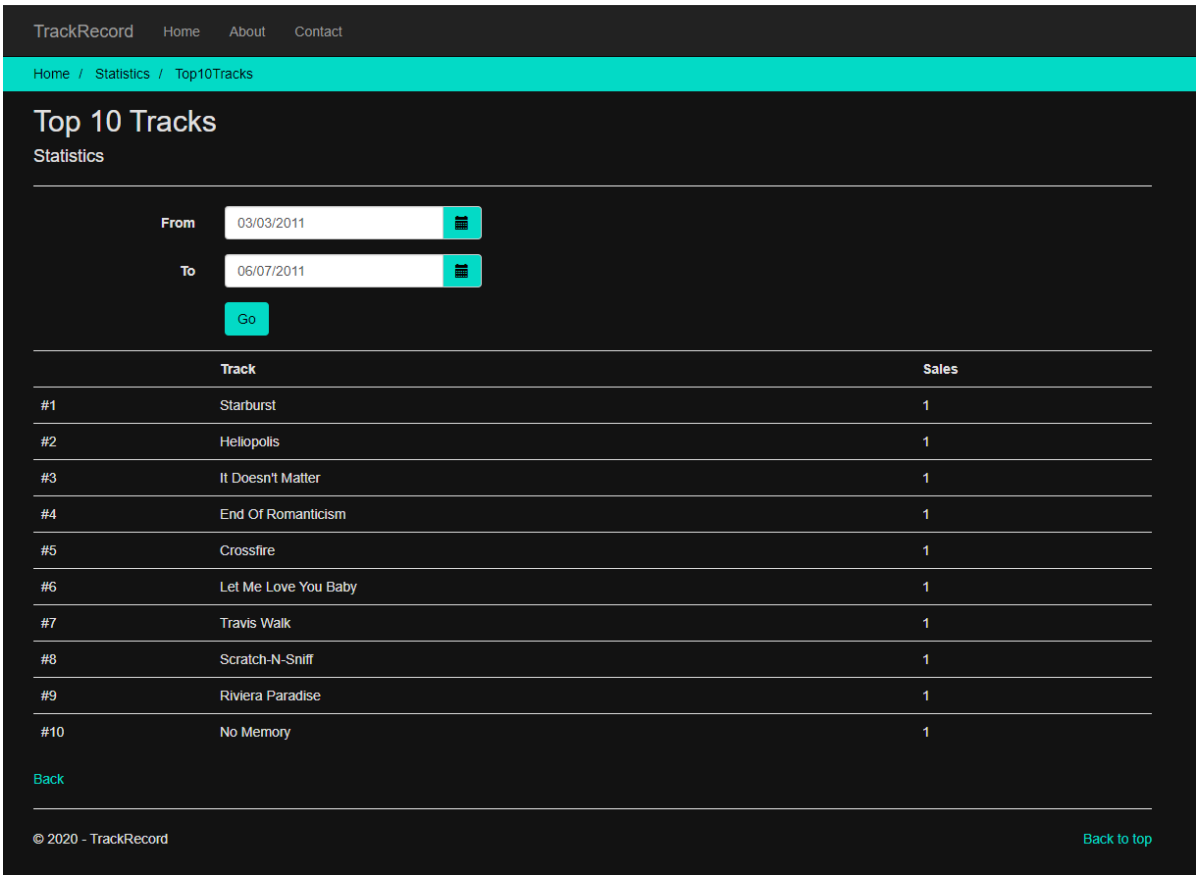
4.2 Κορυφαία κομμάτια

Ο χρήστης μπορεί να δει ποια είναι τα 10 κορυφαία κομμάτια σε πωλήσεις για ένα χρονικό διάστημα της επιλογής του. Ο χρήστης συμπληρώνει το χρονικό διάστημα που επιθυμεί στη φόρμα που φαίνεται στην Εικόνα 30.



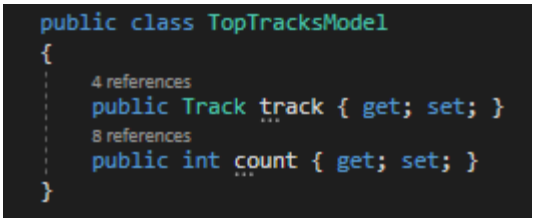
Εικόνα 30 - Φόρμα ανάκτησης 10 κορυφαίων κομματιών

Όταν ο χρήστης πατήσει το κουμπί "Go", καλείται η συνάρτηση GetTracks, η οποία βρίσκεται στον StatisticsController. Η συνάρτηση βρίσκει μέσω του πίνακα InvoiceLines πόσες φορές έχουν πουληθεί τα διάφορα κομμάτια,λαμβάνοντας υπόψη μόνο τις πωλήσεις που έχουν γίνει μέσα στο χρονικό διάστημα που έχει ορίσει ο χρήστης. Τα κομμάτια ταξινομούνται με βάση τις πωλήσεις και επιλέγονται τα πρώτα 10 για προβολή στο χρήστη. Στην Εικόνα 31 φαίνεται το αποτέλεσμα αυτής της διαδικασίας. Τα αποτελέσματα εμφανίζονται σε μορφή λίστας. Στην πρώτη στήλη εμφανίζεται η κατάταξη του κομματιού. Στη δεύτερη στήλη εμφανίζεται το όνομα του κομματιού. Στην τρίτη στήλη εμφανίζονται οι πωλήσεις του κομματιού.



Εικόνα 31 - Αποτέλεσμα ανάκτησης κορυφαίων 10 κομματιών για το χρονικό διάστημα 03/03/2011 - 06/07/2011.

Η συνάρτηση GetTracks δεν περνά τα κομμάτια στη View ως μια λίστα από αντικείμενα τύπου Track. Για αυτό το σκοπό, υπάρχει το Model TopTracksModel, το οποίο έχει ως properties ένα αντικείμενο τύπου Track κι έναν αριθμό, ο οποίος αντιπροσωπεύει τον αριθμό των πωλήσεων του κομματιού.

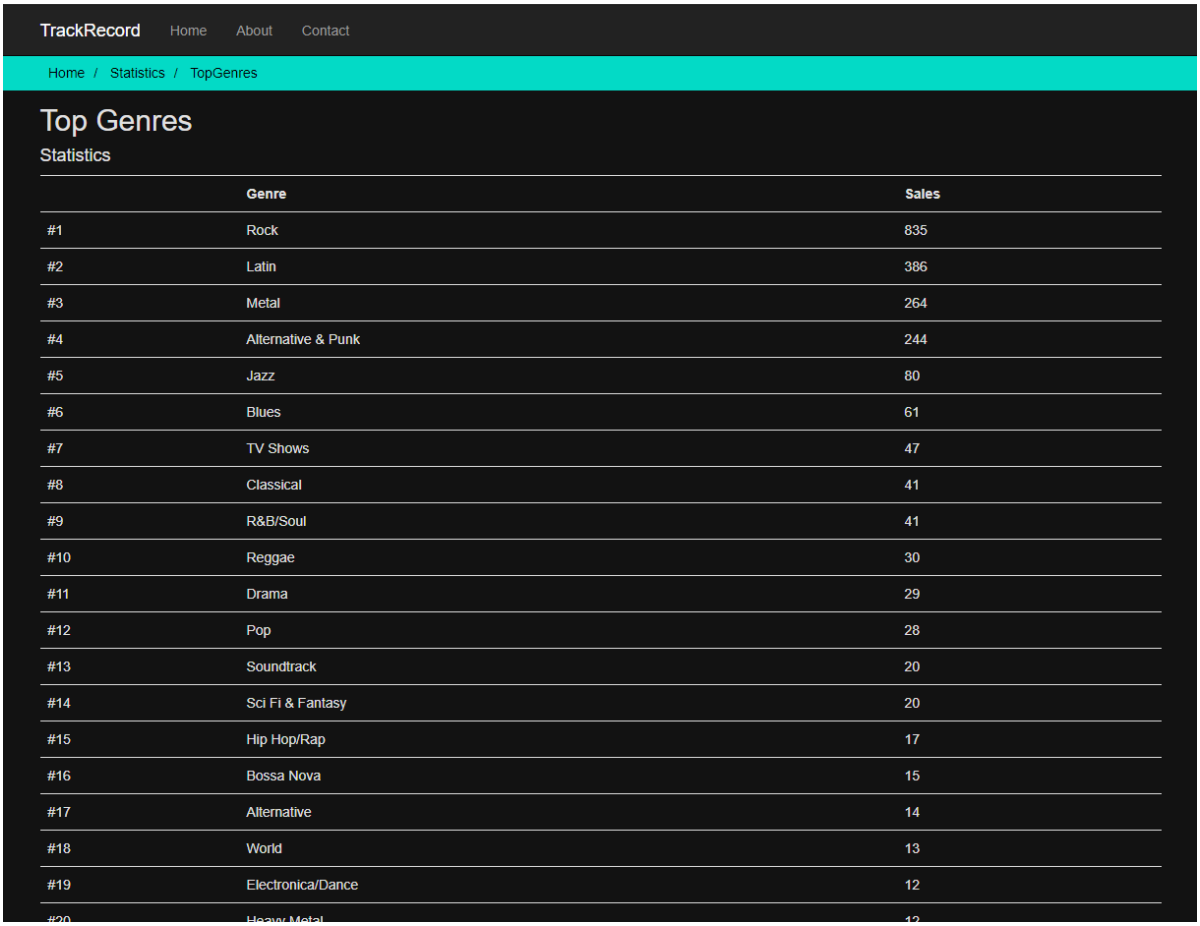


Εικόνα 32 - TopTracksModel

4.3 Κορυφαία είδη

Ο χρήστης μπορεί να δει ποια είναι διαχρονικά τα κορυφαία είδη στις προτιμήσεις των πελατών. Σε αντίθεση με τα άλλα στατιστικά, σε αυτή την περίπτωση ο χρήστης δε μπορεί να παραμετροποιήσει την εξαγωγή. Η εξαγωγή γίνεται μέσω της συνάρτησης GetGenres στον StatisticsController. Η συνάρτηση ομαδοποιεί τα αντικείμενα του πίνακα InvoiceLines, με βάση το είδος του τραγουδιού τους. Δημιουργεί και ταξινομεί μια λίστα αντικειμένων τύπου TopGenresModel, ένα Model το οποίο έχει ως properties ένα αντικείμενο τύπου Genre κι έναν αριθμό, ο οποίος αντιπροσωπεύει τις πωλήσεις κομματιών του είδους. Το αποτέλεσμα φαίνεται στην Εικόνα 34. Τα είδη εμφανίζονται σε μορφή λίστας, όπου σε κάθε γραμμή φαίνεται η κατάταξη του είδους, το όνομά του και ο αριθμός των κομματιών του είδους που πουλήθηκαν.

Εικόνα 33 - Συνάρτηση TopGenres



	Genre	Sales
#1	Rock	835
#2	Latin	386
#3	Metal	264
#4	Alternative & Punk	244
#5	Jazz	80
#6	Blues	61
#7	TV Shows	47
#8	Classical	41
#9	R&B/Soul	41
#10	Reggae	30
#11	Drama	29
#12	Pop	28
#13	Soundtrack	20
#14	Sci Fi & Fantasy	20
#15	Hip Hop/Rap	17
#16	Bossa Nova	15
#17	Alternative	14
#18	World	13
#19	Electronica/Dance	12
#20	Heavy Metal	12

Εικόνα 34 - Κορυφαία είδη στις προτιμήσεις των πελατών