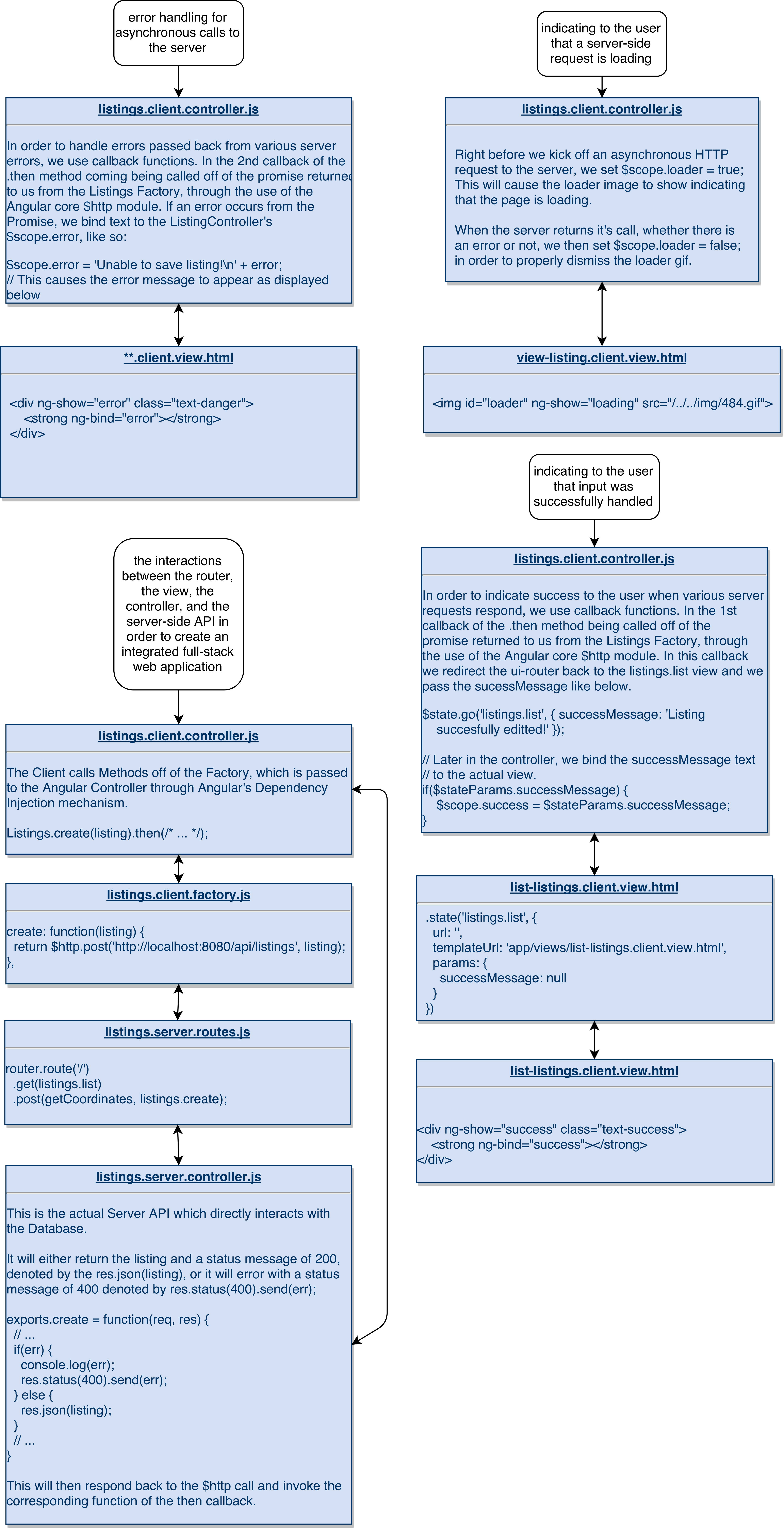

listings.client.controller.js

In order to handle errors passed back from various server errors, we use callback functions. In the 2nd callback of the .then method coming being called off of the promise returned to us from the Listings Factory, through the use of the Angular core $http module. If an error occurs from the Promise, we bind text to the ListingController's $scope.error, like so:

```
$scope.error = 'Unable to save listing!\n' + error;
// This causes the error message to appear as displayed below
```

****.client.view.html**

```
<div ng-show="error" class="text-danger">
    <strong ng-bind="error"></strong>
</div>
```

indicating to the user that a server-side request is loading

listings.client.controller.js

Right before we kick off an asynchronous HTTP request to the server, we set $scope.loader = true; This will cause the loader image to show indicating that the page is loading.

When the server returns it's call, whether there is an error or not, we then set $scope.loader = false; in order to properly dismiss the loader gif.

view-listing.client.view.html

```
<img id="loader" ng-show="loading" src="/../../img/484.gif">
```

listings.client.controller.js

In order to indicate success to the user when various server requests respond, we use callback functions. In the 1st callback of the .then method being called off of the promise returned to us from the Listings Factory, through the use of the Angular core $http module. In this callback we redirect the ui-router back to the listings.list view and we pass the sucessMessage like below.

```
$state.go('listings.list', { successMessage: 'Listing
    succesfully editted!' });

// Later in the controller, we bind the successMessage text
// to the actual view.
if($stateParams.successMessage) {
    $scope.success = $stateParams.successMessage;
}
```

list-listings.client.view.html

```
 .state('listings.list', {
   url: '',
   templateUrl: 'app/views/list-listings.client.view.html',
   params: {
     successMessage: null
   }
 })
```

list-listings.client.view.html

```
<div ng-show="success" class="text-success">
    <strong ng-bind="success"></strong>
</div>
```

the interactions between the router, the view, the controller, and the server-side API in order to create an integrated full-stack web application

listings.client.controller.js

The Client calls Methods off of the Factory, which is passed to the Angular Controller through Angular's Dependency Injection mechanism.

```
Listings.create(listing).then(/* ... */);
```

listings.client.factory.js

```
create: function(listing) {
  return $http.post('http://localhost:8080/api/listings', listing);
},
```

listings.server.routes.js

```
router.route('/')
  .get(listings.list)
  .post(getCoordinates, listings.create);
```

listings.server.controller.js

This is the actual Server API which directly interacts with the Database.

It will either return the listing and a status message of 200, denoted by the res.json(listing), or it will error with a status message of 400 denoted by res.status(400).send(err);

```
exports.create = function(req, res) {
  // ...
  if(err) {
    console.log(err);
    res.status(400).send(err);
  } else {
    res.json(listing);
  }
  // ...
}
```

This will then respond back to the $http call and invoke the corresponding function of the then callback.