

JWT In Android

donation-web

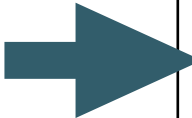
donation-web - Open these APIs

```
{ method: 'GET', path: '/api/candidates', config: CandidatesApi.find },
```



```
exports.find = {  
  auth: false,  
  handler: function (request, reply) {  
    Candidate.find({}).exec().then(candidates => {  
      reply(candidates);  
    }).catch(err => {  
      reply(Boom.badImplementation('error accessing db'));  
    });  
  },  
};
```

```
{ method: 'POST', path: '/api/users', config: UsersApi.create },
```



```
exports.create = {  
  auth: false,  
  handler: function (request, reply) {  
    const user = new User(request.payload);  
    user.save().then(newUser => {  
      reply(newUser).code(201);  
    }).catch(err => {  
      reply(Boom.badImplementation('error creating User'));  
    });  
  },  
};
```

Return user details for authenticated user

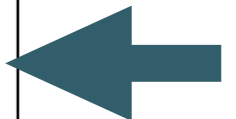
```
exports.authenticate = {  
  
  auth: false,  
  
  handler: function (request, reply) {  
    const user = request.payload;  
    User.findOne({ email: user.email }).then(foundUser => {  
      if (foundUser && foundUser.password === user.password) {  
        const token = utils.createToken(foundUser);  
        reply({ success: true, token: token, user: foundUser }).code(201);  
      } else {  
        reply({ success: false, message: 'Authentication failed. User not found.' }).code(201);  
      }  
    }).catch(err => {  
      reply(Boom.notFound('internal db failure'));  
    });  
  },  
  
};
```



Make Donation Update

- Fully populate new donation before returning to client

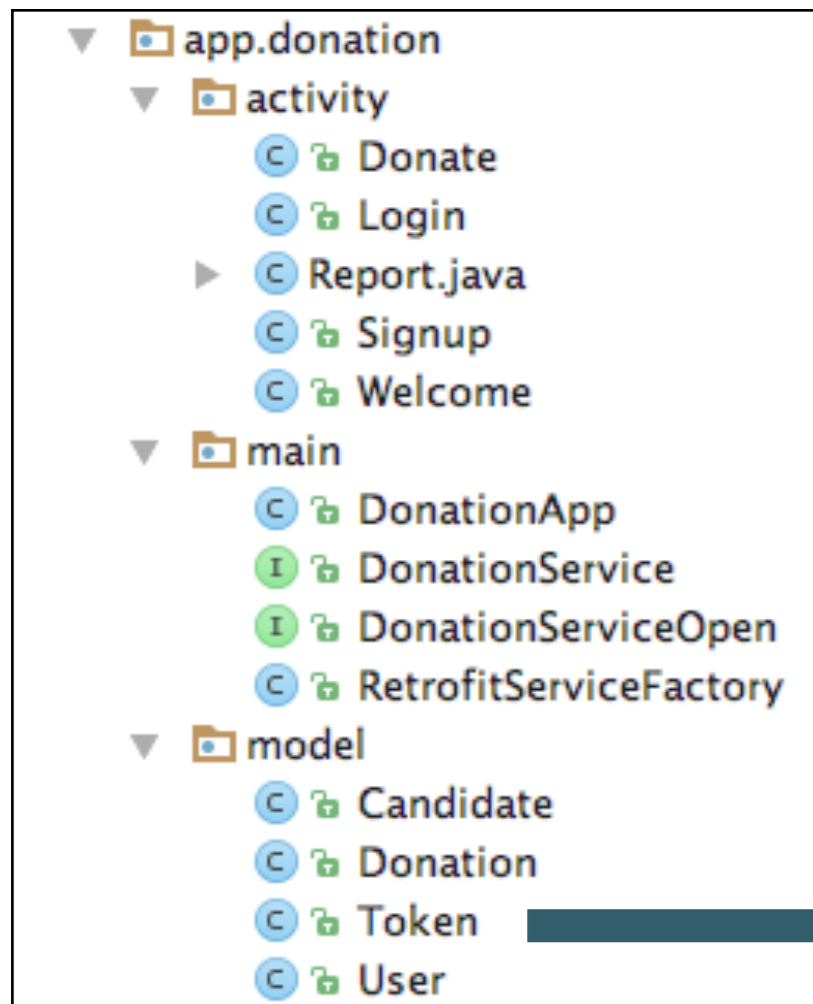
```
exports.makeDonation = {  
  
  auth: {  
    strategy: 'jwt',  
  },  
  
  handler: function (request, reply) {  
    const donation = new Donation(request.payload);  
    donation.candidate = request.params.id;  
    donation.donor = utils.getUserIdFromRequest(request);  
    donation.save().then(newDonation => {  
      return Donation.findOne(newDonation).populate('candidate').populate('donor');  
    }).then(newDonation => {  
      reply(newDonation).code(201);  
    }).catch(err => {  
      reply(Boom.badImplementation('error making donation'));  
    });  
  },  
  
};
```



donation-android

Token donation-web

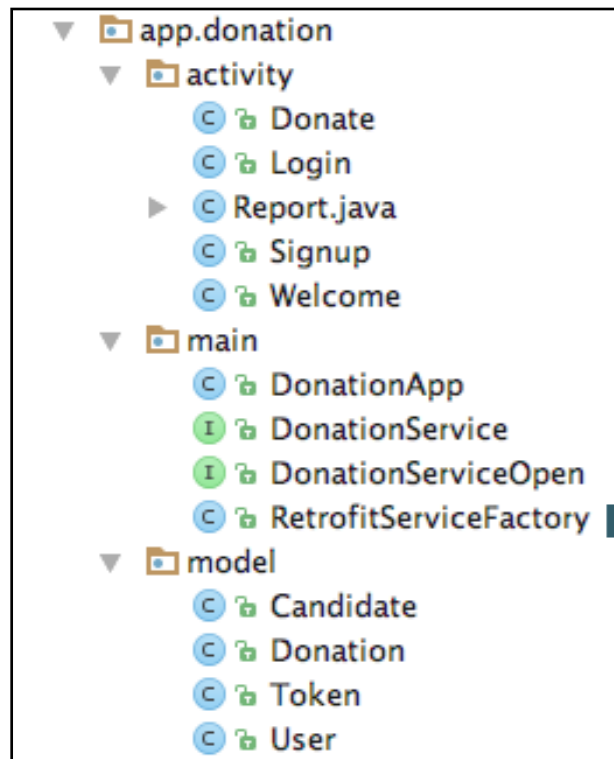
```
user = request.payload;  
findOne({ email: user.email }).then(foundUser => {  
  foundUser && foundUser.password === user.password) {  
    let token = utils.createToken(foundUser);  
    res.json({ success: true, token: token, user: foundUser }).code(200);  
  } else {  
    res.json({ success: false, message: 'Authentication failed. User not found.' }).code(401);  
  }  
}  
if (err) {  
  res.json({ success: false, message: 'internal db failure' });  
}
```



- Client model for token retrieved from service

```
public class Token  
{  
  public boolean success;  
  public String token;  
  public User user;  
  
  public Token(boolean success, String token)  
  {  
    this.success = success;  
    this.token = token;  
  }  
}
```

RetrofitServiceFactory



```
public class RetrofitServiceFactory
{
    public static final String API_BASE_URL = "http://10.0.2.2:4000";

    private static OkHttpClient.Builder httpClient = new OkHttpClient.Builder();

    private static Builder
        builder = new Builder().baseUrl(API_BASE_URL)
            .addConverterFactory(GsonConverterFactory.create());

    public static <S> S createService(Class<S> serviceClass)
    {
        ...
    }

    public static <S> S createService(Class<S> serviceClass, final String authToken)
    {
        ...
    }
}
```

- Factory class to create the service proxies in android.
- Two factory methods:
 - Open api access
 - JWT secured api access

Open API Access

```
public class RetrofitServiceFactory
{
    ...

    public static <S> S createService(Class<S> serviceClass)
    {
        Retrofit retrofit = builder.client(httpClient.build()).build();
        return retrofit.create(serviceClass);
    }
    ...
}
```

```
public interface DonationServiceOpen
{
    @GET("/api/candidates")
    Call<List<Candidate>> getAllCandidates();

    @POST("/api/users")
    Call<User> createUser(@Body User user);

    @POST("/api/users/authenticate")
    Call<Token> authenticate(@Body User user);
}
```

```
donationServiceOpen = RetrofitServiceFactory.createService(DonationServiceOpen.class);
```

- Create the proxy in DonationApp

- Factory method to create proxy for open api
- Interface for open api

Secured API Access

jwt



```
public static <S> S createService(Class<S> serviceClass, final String authToken)
{
    if (authToken != null)
    {
        httpClient.addInterceptor(new Interceptor()
        {
            @Override
            public Response intercept(Interceptor.Chain chain) throws IOException
            {
                Request original = chain.request();
                Request.Builder requestBuilder = original.newBuilder()
                    .header("Authorization", "bearer " + authToken)
                    .method(original.method(), original.body());

                Request request = requestBuilder.build();
                return chain.proceed(request);
            }
        });
    }

    OkHttpClient client = httpClient.build();
    Retrofit retrofit = builder.client(client).build();
    return retrofit.create(serviceClass);
}
```

- Factory method to create proxy for secured api

```
public interface DonationService
{
    @GET("/api/donations")
    Call<List<Donation>> getAllDonations();

    @POST("/api/candidates/{id}/donations")
    Call<Donation> createDonation(@Path("id") String id, @Body Donation donation);
}
```

- Interface for secured api

Creating the Secured API Proxy

```
public boolean validUser (String email, String password)
{
    User user = new User ("", "", email, password);
    donationServiceOpen.authenticate(user);
    Call<Token> call = (Call<Token>) donationServiceOpen.authenticate (user);
    call.enqueue(this);
    return true;
}

@Override
public void onResponse(Call<Token> call, Response<Token> response) {
    Token auth = response.body();
    currentUser = auth.user;
    donationService = RetrofitServiceFactory.createService(DonationService.class, auth.token);
    Log.v("Donation", "Authenticated " + currentUser.firstName + ' ' + currentUser.lastName);
}
```

- Create secured proxy only when a user is successfully authenticated