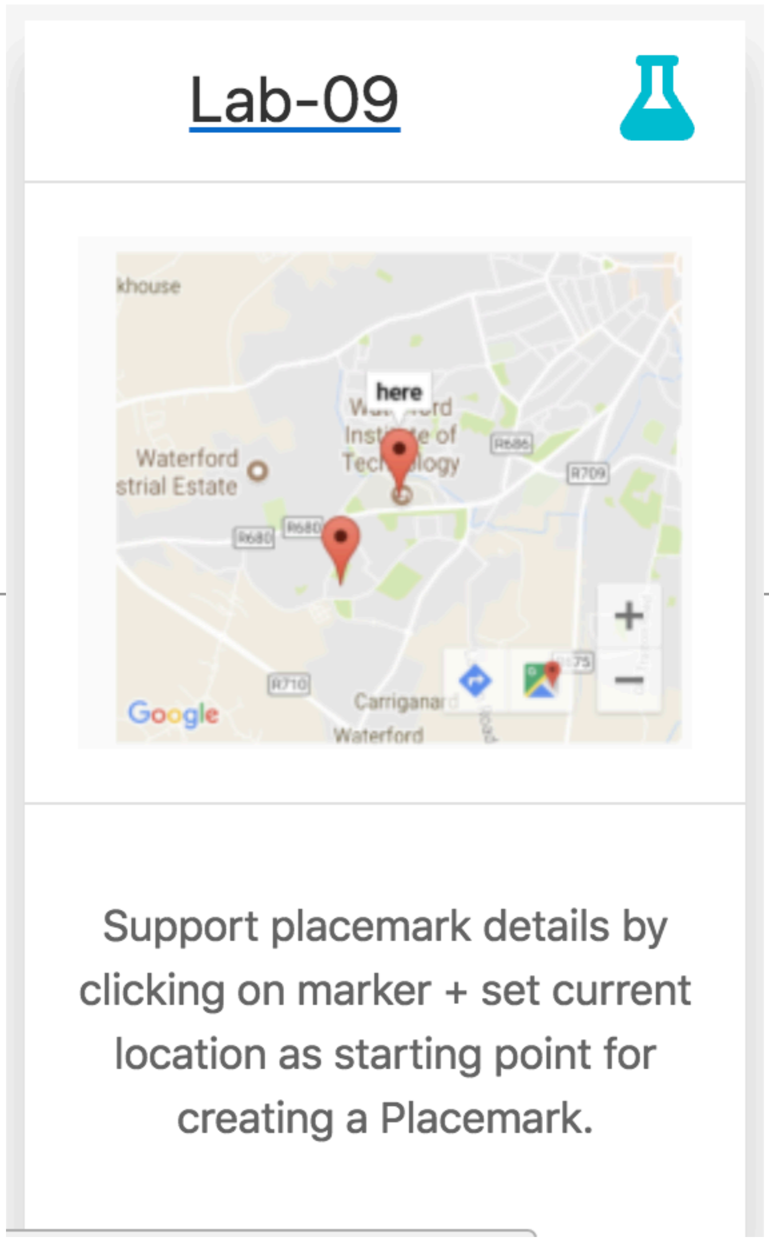
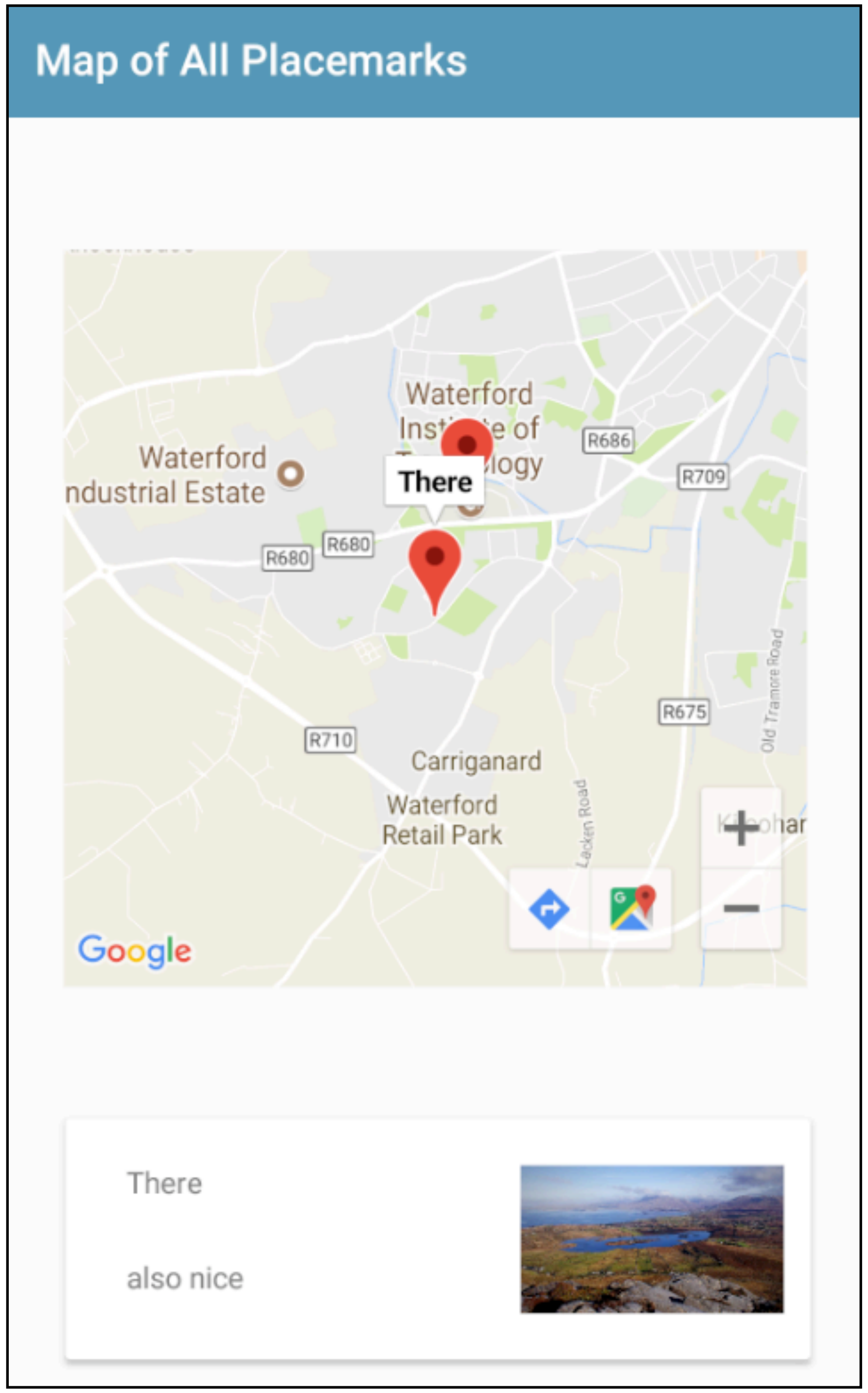
Maps - Markers



Lab-09

Support placemark details by clicking on marker + set current location as starting point for creating a Placemark.

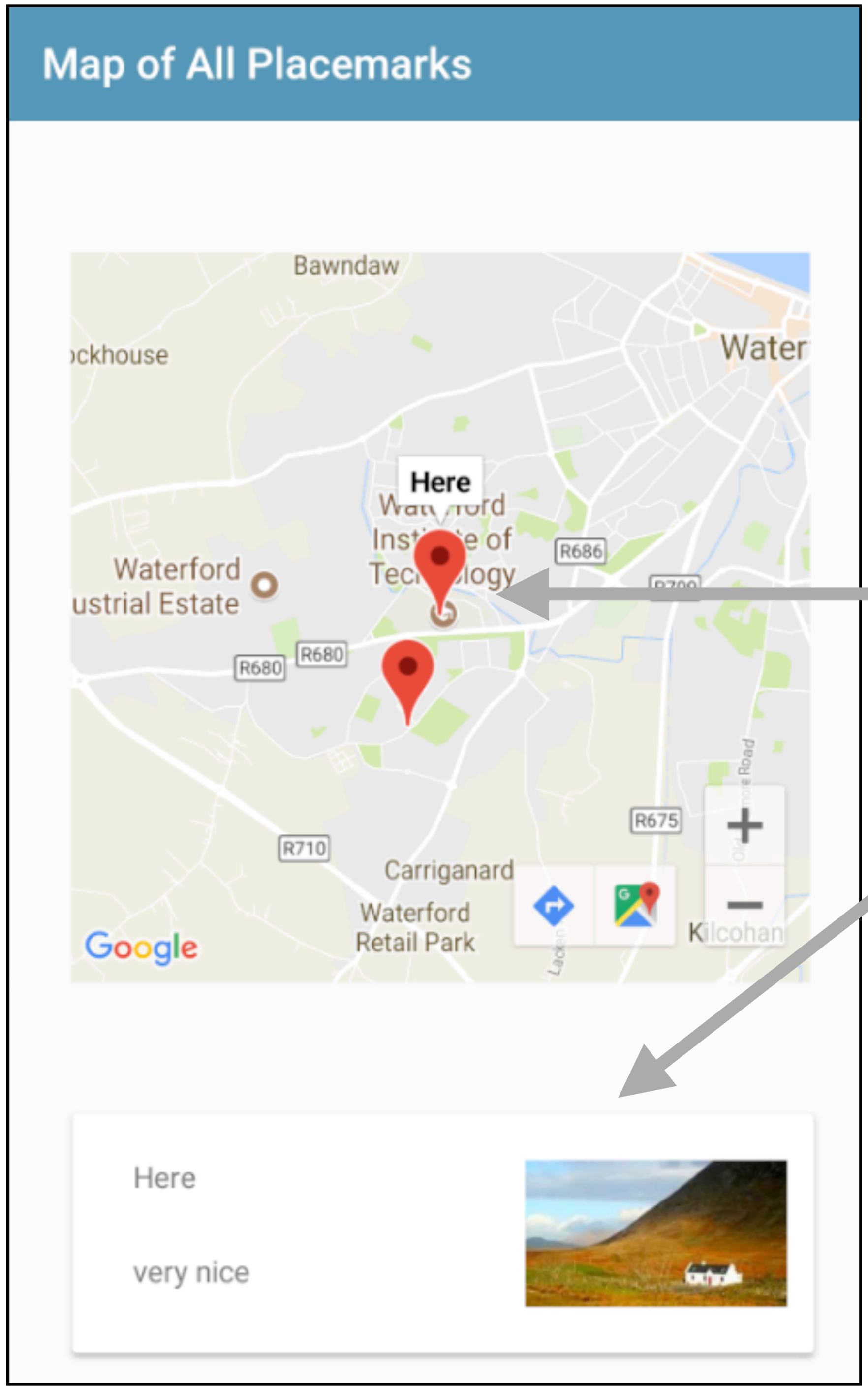
PlacemarkMapsActivity



Map of All Placemarks

There

also nice



Map of All Placemarks

Here

very nice

Select placemark…

dynamically updates card details

PlacemarkStore - findById

```
interface PlacemarkStore {
  suspend fun findAll(): List<PlacemarkModel>
  suspend fun findById(id:Long) : PlacemarkModel?
  fun create(placemark: PlacemarkModel)
  fun update(placemark: PlacemarkModel)
  fun delete(placemark: PlacemarkModel)
}
```

new method to search for placemaker by id

PlacemarkDao- findById

```
@Dao
interface PlacemarkDao {

  @Insert(onConflict = OnConflictStrategy.REPLACE)
  fun create(placemark: PlacemarkModel)

  @Query("SELECT * FROM PlacemarkModel")
  fun findAll(): List<PlacemarkModel>

  @Query("select * from PlacemarkModel where id = :id")
  fun findById(id: Long): PlacemarkModel

  @Update
  fun update(placemark: PlacemarkModel)

  @Delete
  fun deletePlacemark(placemark: PlacemarkModel)
}
```

new method to search for placemaker by id

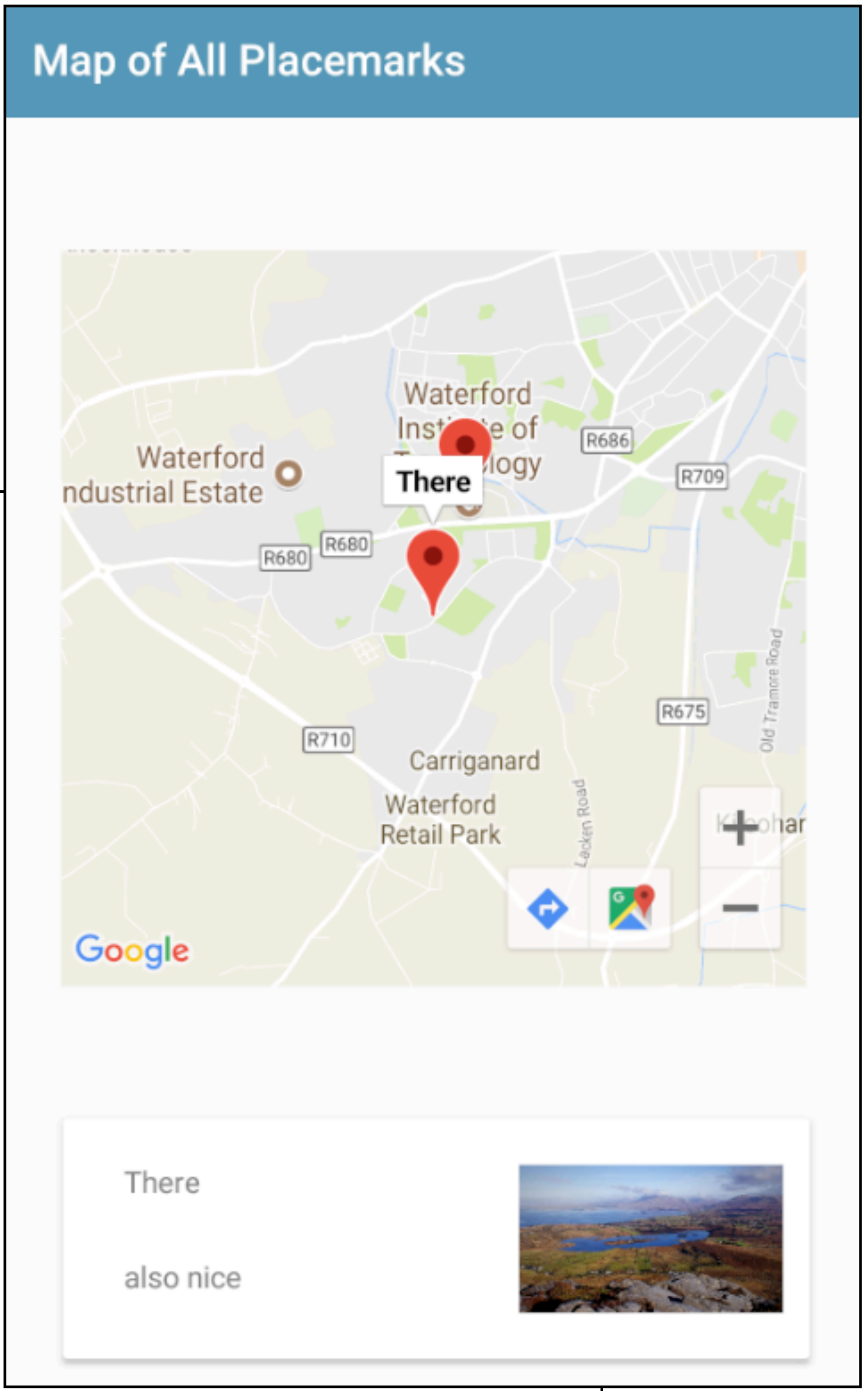
PlacemarkStoreRoom

```kotlin
class PlacemarkStoreRoom(val context: Context) : PlacemarkStore {

  var dao: PlacemarkDao


  …

  override suspend fun findById(id: Long): PlacemarkModel? {
    val deferredPlacemark = bg {
      dao.findById(id)
    }
    val placemark = deferredPlacemark.await()
    return placemark
  }
}
```

new method to search for placemaker by id

PlacemarkMapsActivity

```kotlin
fun configureMap() {
  map.uiSettings.setZoomControlsEnabled(true)
  map.setOnMarkerClickListener(this)
  async(UI) {
    app.placemarks.findAll().forEach {
      val loc = LatLng(it.lat, it.lng)
      val options = MarkerOptions().title(it.title).position(loc)
      map.addMarker(options).tag = it.id
      map.moveCamera(CameraUpdateFactory.newLatLngZoom(loc, it.zoom))
    }
  }
}

override fun onMarkerClick(marker: Marker): Boolean {
  async(UI) {
    val tag = marker.tag as Long
    val placemark = app.placemarks.findById(tag)
    currentTitle.text = placemark!!.title
    currentDescription.text = placemark!!.description
    imageView.setImageBitmap(readImageFromPath(this@PlacemarkMapsActivity, placemark.image))
  }
  return false
}
```

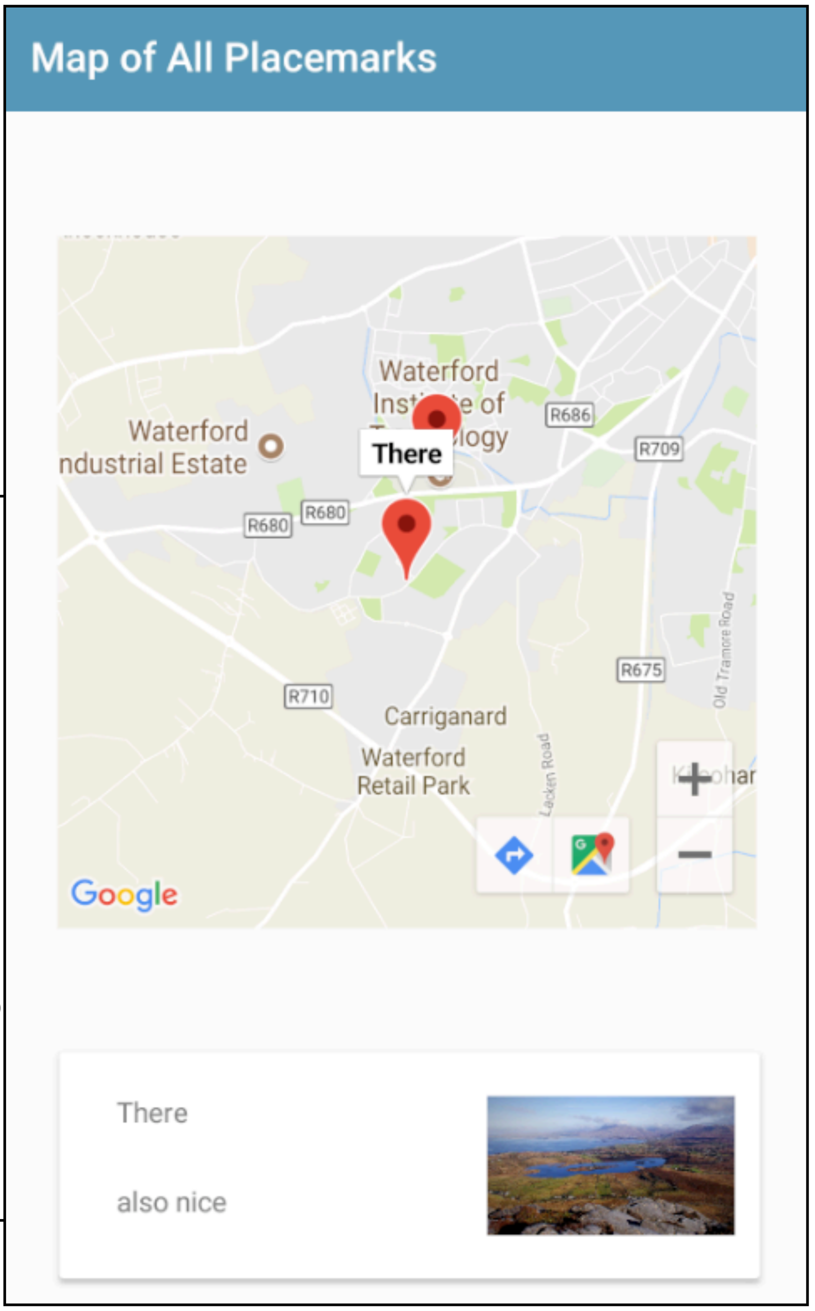

Map of All Placemarks

There

also nice

PlacemarkMapsActivity

```
fun configureMap() {
  map.uiSettings.setZoomControlsEnabled(true)
  map.setOnMarkerClickListener(this)
  async(UI) {
    app.placemarks.findAll().forEach {
      val loc = LatLng(it.lat, it.lng)
      val options = MarkerOptions().title(it.title).position(loc)
      map.addMarker(options).tag = it.id
      map.moveCamera(CameraUpdateFactory.newLatLngZoom(loc, it.zoom))
    }
  }
}
```



There

also nice

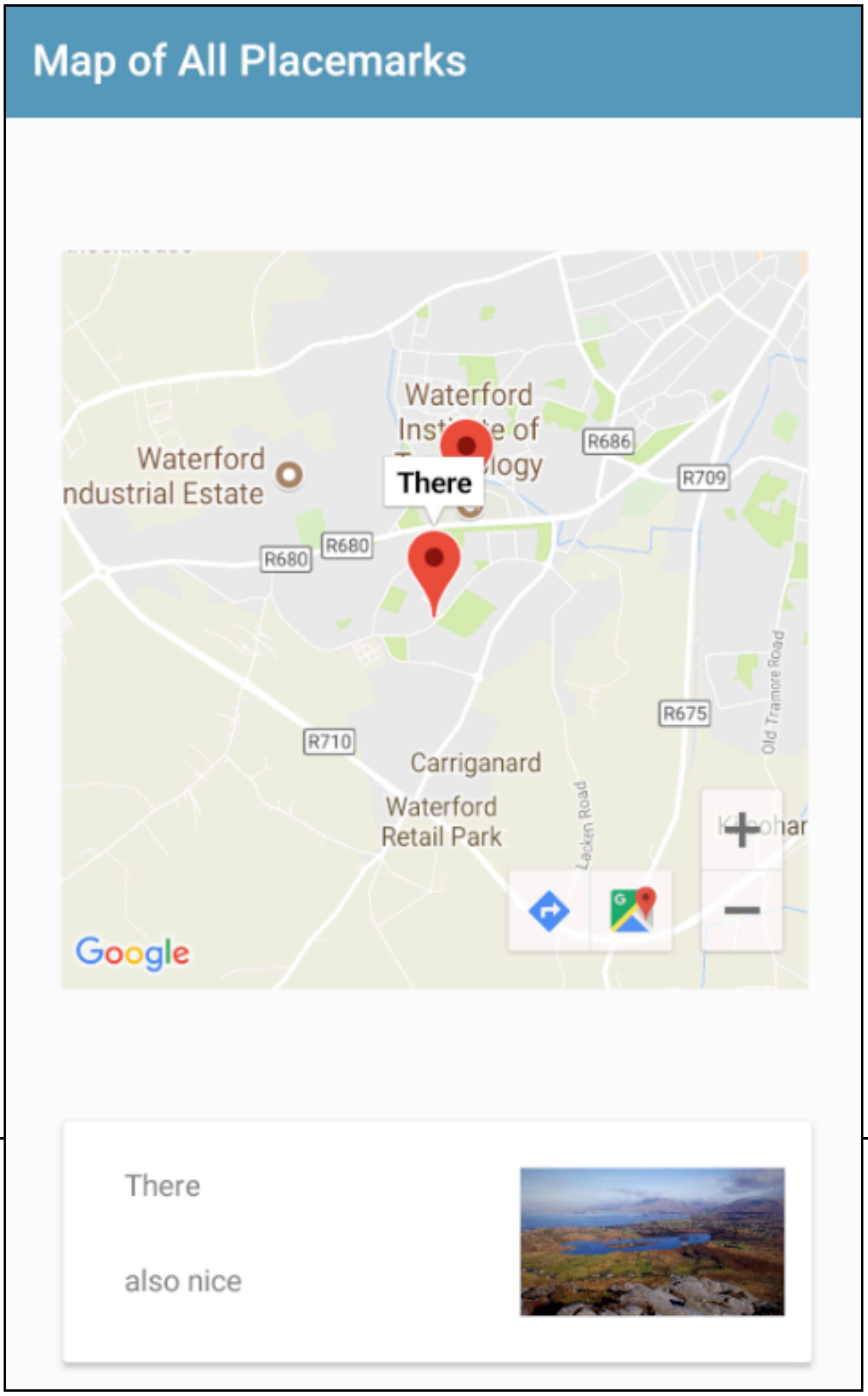
Add all placemarks to map
As each placemark is added - include its **ID** as a **TAG**

```
map.addMarker(options).tag = it.id
```

PlacemarkMapsActivity

When a marker clicked -

- recover the marked tag
- look up tag in placemark database
- put placemark details in card



There

also nice

```
override fun onMarkerClick(marker: Marker): Boolean {
  async(UI) {
    val tag = marker.tag as Long
    val placemark = app.placemarks.findById(tag)
    currentTitle.text = placemark!!.title
    currentDescription.text = placemark!!.description
    imageView.setImageBitmap(readImageFromPath(this@PlacemarkMapsActivity, placemark.image))
  }
  return false
}
```