## Exercise 6.1: Perform API calls in background

Change API calls to occur in the background!

Take the following steps:

* Remove the lines that start with *StrictMode...* from the *onCreate()* method of your activities.
* Create classes for each API call that implement the interface runnable.
* Move the methods that make API calls from the activity classes to the corresponding Runnable class. If needed create a constructor that can take references to required objects (f.e. activity) and adjust the code as necessary.
* Implement the *run()* method of your runnables.
* When the users select the menu entry to refresh the recipe list, start this process in the background by instantiating FetchRecipeListRunnable and starting a new thread.
* Ask yourself: What would happen, when the user starts a refresh while one is still in progress?

When in doubt, ask the advisor!

Do you notice that the UI stays responsive during the update now?

**Note:** You can simulate a “bad” network connection in the emulator by opening the “Extended controls” (button “…” in the toolbar) and then in the area “Cellular” selecting a worse network type or signal strength.

## Exercise 6.2: Add toast messages

* Create toasts that informs the user about the result of the executed HTTP requests. Do it for the cases where you think it would be suitable (f.e. error messages).
* The toasts are triggered from the *run()* method once the API call has been completed.
* However, a toast can only be started from the UI thread. This means you will need to create a new runnable that is passed to the UI thread for execution:

**activity**.runOnUiThread(**new** Runnable() {  
 @Override  
 **public void** run() {  
 // Create and show toast!

}  
});

## Exercise 6.3: A Notification for new recipes (optional)

Show an additional notification once the refresh has finished!

## Exercise 6.4: Edit a recipe via the Rest API (optional)

* Add a new menu entry to the RecipeDetailsActivity menu that allows you to edit the currently shown recipe.
* The fetch details endpoint returns an isCreator variable, which determines if the user has the permission to update or delete the *Recipe*. Make the new menu entry visible only if isCreator is true.
* Make the Recipe class implement the Serializable interface and pass the Recipe object to the SaveRecipeActivityas an extra*.*
* Change the SaveRecipeActivity and the corresponding runnable to load the recipe data inside the form components if provided. Use the PUT endpoint to update the recipe.

## Exercise 6.5: Delete a recipe via the Rest API (optional)

* Add a new menu entry to the RecipeDetailsActivity menu that allows you to delete the currently shown recipe. Make the new menu entry visible only if isCreator is true.
* You will need to create a new runnable for the tasks. Use the DELETE endpoint.
* Make sure you call *finish()* on the RecipeDetailsActivity upon successful deletion. Optionally, you can show a toast message as confirmation to the user that operation was successful.

## Exercise 6.6: Find out how to deploy your app

Create a signed APK as shown during the lecture.