

Design Patterns

MSc in Computer Science

Produced
by

Eamonn de Leastar (edeleastar@wit.ie)

Department of Computing, Maths & Physics
Waterford Institute of Technology

<http://www.wit.ie>

<http://elearning.wit.ie>



Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRCE



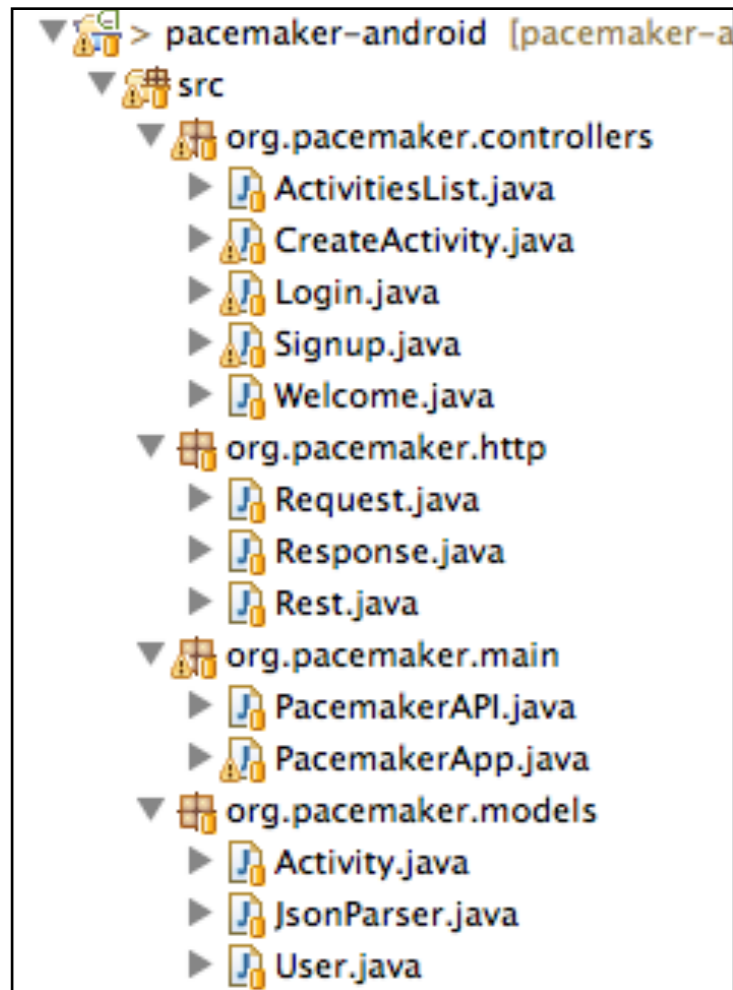
Pacemaker V3 - Lab-08

Half Sync / Half Async Case Study

- Android Activities are synchronous - single threaded in the context of user interaction
- Service access is asynchronous - inherently unreliable access to remote application service
- Half Sync/Half Async an appropriate pattern to tackle this problem

- Uses pacemaker-service

Pacemaker V4 - Lab 09



- **controllers**

- display and handle all UI

- **http**

- General purpose classes to support asynchronous http request/response to/from donation-service

- **main**

- facade for model / http interactions and sync capability

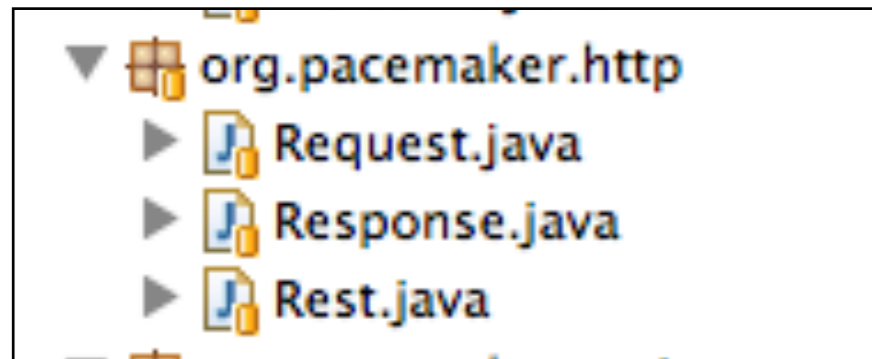
- **models**

- Local copies of core information models for the application (download from pacemaker-play)
- Parsers (transformers) for converting objects into format suitable for upload/download to/from pacemaker-service

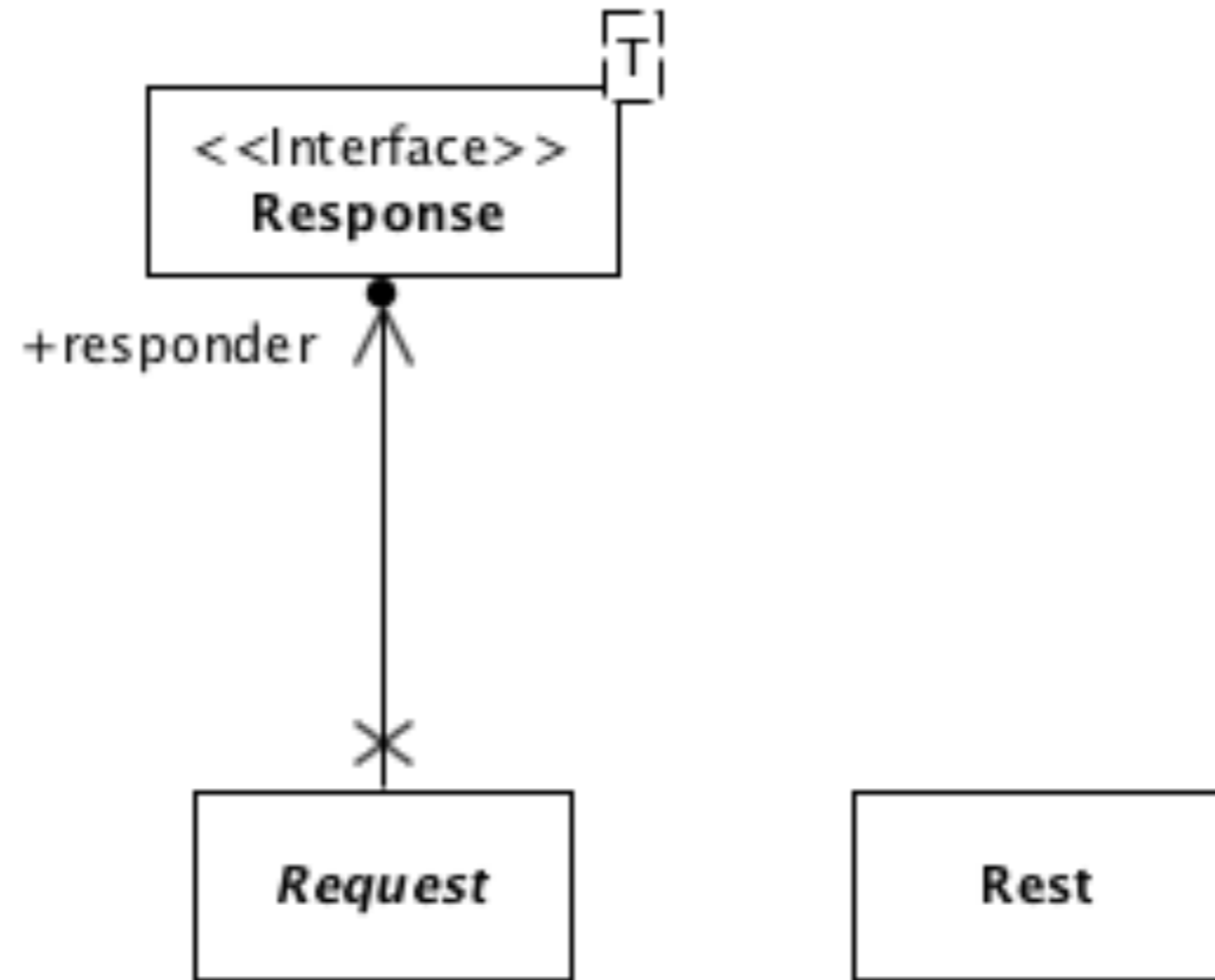
Android AsyncTask Class

- AsyncTask allows you to perform asynchronous work on your user interface.
- It performs the blocking operations in a **worker thread** and then publishes the results on the UI thread.
- You subclass AsyncTask and implement the doInBackground() callback method, which runs in a pool of background threads.
- To update your UI, you implement onPostExecute(), which delivers the result from doInBackground() and runs in the UI thread, so you can safely update your UI.

HTTP




- **http**
 - General purpose classes to support **asynchronous** http request/response to/from donation-service
 - These requests are performed in a separate thread of execution



This server is your development machine

Rest

- Issue HTTP requests to a server
 - GET
 - DELETE
 - PUT
 - POST
- (http verbs)



```
public class Rest
{
    private static final String URL = "http://10.0.2.2:9000";

    public static String get(String path) throws Exception
    {
        //...
    }

    public static String delete(String path)
    {
        //...
    }

    public static String put(String path, String json)
    {
        //...
    }

    public static String post(String path, String json)
    {
        //...
    }
}
```


Rest

```
public class Rest
{
    private static final String URL = "http://10.0.2.2:9000";

    public static String get(String path) throws Exception
    {
        //...
    }

    public static String delete(String path) throws Exception
    {
        //...
    }

    public static String put(String path, String json) throws Exception
    {
        //...
    }

    public static String post(String path, String json) throws Exception
    {
        //...
    }
}
```

- Rest class can only send/receive strings (no Model objects like User or Donation)
- Assumes all strings are Json encoded
- Will very likely throw 'exceptions' if server error, network problem or other related issue
- No need to edit/maintain this class as it adheres to HTTP protocol conventions
- Is independent of donation application, and can be used in other apps as is

Response

- An Interface that must be implemented by the Activity that initiated the request.
- Is 'paramaterised' by T, which will typically be some model object we are requesting/updating
 - e.g. User, Donation
- However, interface is application independent, and can be used in other applications not related to Donation app.

```
public interface Response<T>
{
    public void setReponse(List<T> aList);
    public void setReponse(T anObject);
    public void errorOccurred (Exception e);
}
```

Callbacks

```
public interface Response<T>
{
    public void setReponse(List<T> aList);
    public void setReponse(T anObject);
    public void errorOccurred (Exception e);
}
```

- When a request is made by an activity, then the activity will be ‘called back’ when a result becomes available.
- One of these three methods will be called:
 - A single object of type T is returned from the service
 - A list of T objects is returned
 - An error has occurred
- The callback will occur on the UI Thread, so the activity can update its components safely

Request

- Put up a dialog saying 'Processing...'
- Launch a background thread/task
- Report when finished to callback
- Reusable class, can be used in apps unrelated to donation.

```
public abstract class Request extends AsyncTask<Object, Void, Object>
{
    //...

    public Request(Context context, Response responder, String message)
    {
        //...
    }

    @Override
    protected void onPreExecute()
    {
        //...
    }

    @Override
    protected Object doInBackground(Object... params)
    {
        //...
    }

    protected abstract Object doRequest(Object... params) throws Exception;

    @Override
    protected void onPostExecute(Object result)
    {
        //...
    }
}
```

Request

```
public abstract class Request extends AsyncTask<Object, Void, Object>
{
    //...

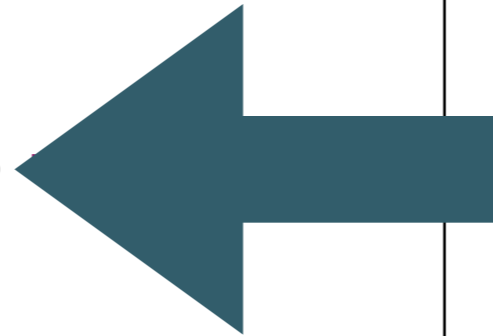
    public Request(Context context, Response responder, String message)
    {
        //...
    }

    @Override
    protected void onPreExecute()
    {
        //...
    }

    @Override
    protected Object doInBackground(Object... params)
    {
        //...
    }

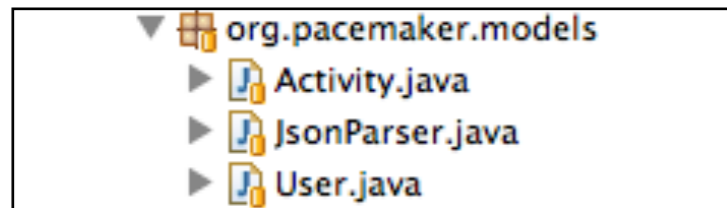
    protected abstract Object doRequest(Object... params)

    @Override
    protected void onPostExecute(Object result)
    {
        //...
    }
}
```



- An 'Abstract' class, so 'abstract' method 'doRequest' must be provided to do the actual background process
- For donation-android app, this will be a call to http.Rest methods to get/set data in android-service

models



- **models**

- Local copies of core information models for the application (download from donation-play)
- Parsers (transformers) for converting objects into format suitable for upload/download to/from donation-service

models

```
public class User
{
    public Long id;
    public String firstname;
    public String lastname;
    public String email;
    public String password;

    public User()
    {}

    public User(String firstname, String lastname, String email, String password)
    {
        this.firstname = firstname;
        this.lastname = lastname;
        this.email = email;
        this.password = password;
    }
}
```

```
public class Activity
{
    public Long id;
    public String type;
    public String location;
    public double distance;

    public Activity()
    {}

    public Activity(String type, String location, double distance)
    {
        this.type = type;
        this.location = location;
        this.distance = distance;
    }
}
```

Parser

- same class as in pacemaker-service
- Convert Model object to/from Json format

```
public class JsonParser
{
    public static JSONSerializer userSerializer      = new JSONSerializer().exclude("class")
                                                    .exclude("persistent")
                                                    .exclude("entityId");
    public static JSONSerializer activitySerializer = new JSONSerializer().exclude("class")
                                                    .exclude("persistent")
                                                    .exclude("entityId");

    public static User json2User(String json)
    {
        return new JSONDeserializer<User>().deserialize(json, User.class);
    }

    public static List<User> json2Users(String json)
    {
        return new JSONDeserializer<ArrayList<User>>().use("values", User.class)
                                                    .deserialize(json);
    }

    public static String user2Json(Object obj)
    {
        return userSerializer.serialize(obj);
    }

    public static Activity json2Activity(String json)
    {
        Activity activity = new JSONDeserializer<Activity>().deserialize(json, Activity.class);
        return activity;
    }

    public static String activity2Json(Object obj)
    {
        return activitySerializer.serialize(obj);
    }

    public static List<Activity> json2Activities (String json)
    {
        return new JSONDeserializer<ArrayList<Activity>>().use("values", Activity.class).deserialize(json);
    }
}
```


PacemakerAPI

- Enable Activities to ‘invoke’ services on pacemaker-service app.
- Specifically:
 - GetUsers
 - GetActivities
 - CreateUser
 - CreateActivity
- Each of these requests is ‘spun-out’ into separate thread

```
public class PacemakerAPI
{
    public static void getUsers(Context context, Response<User> response,
                               String dialogMessage)
    {
        new GetUsers(context, response, dialogMessage).execute();
    }

    public static void createUser(Context context, Response<User> response,
                                   String dialogMessage, User user)
    {
        new CreateUser(context, response, dialogMessage).execute(user);
    }

    public static void getActivities(Context context, User user,
                                     Response<Activity> response, String dialogMessage)
    {
        new GetActivities(context, user, response, dialogMessage).execute();
    }

    public static void createActivity(Context context, User user,
                                      Response<Activity> response,
                                      String dialogMessage, Activity activity)
    {
        new CreateActivity(context, user, response, dialogMessage).execute(activity);
    }
}
```

GetUsers and CreateUser Requests

- The doRequest() methods will run in a background thread
- ... and will use the Rest class to communicate with the server

```
class GetUsers extends Request
{
    public GetUsers(Context context, Response<User> callback, String message)
    {
        super(context, callback, message);
    }

    @Override
    protected List<User> doRequest(Object... params) throws Exception
    {
        String response = Rest.get("/api/users");
        List<User> userList = JsonParsers.json2Users(response);
        return userList;
    }
}

class CreateUser extends Request
{
    public CreateUser(Context context, Response<User> callback, String message)
    {
        super(context, callback, message);
    }

    @Override
    protected User doRequest(Object... params) throws Exception
    {
        String response = Rest.post ("/api/users", JsonParsers.user2Json(params[0]));
        return JsonParsers.json2User(response);
    }
}
```

GetActivities and CreateActivity Requests

```
class GetActivities extends Request
{
    private User user;

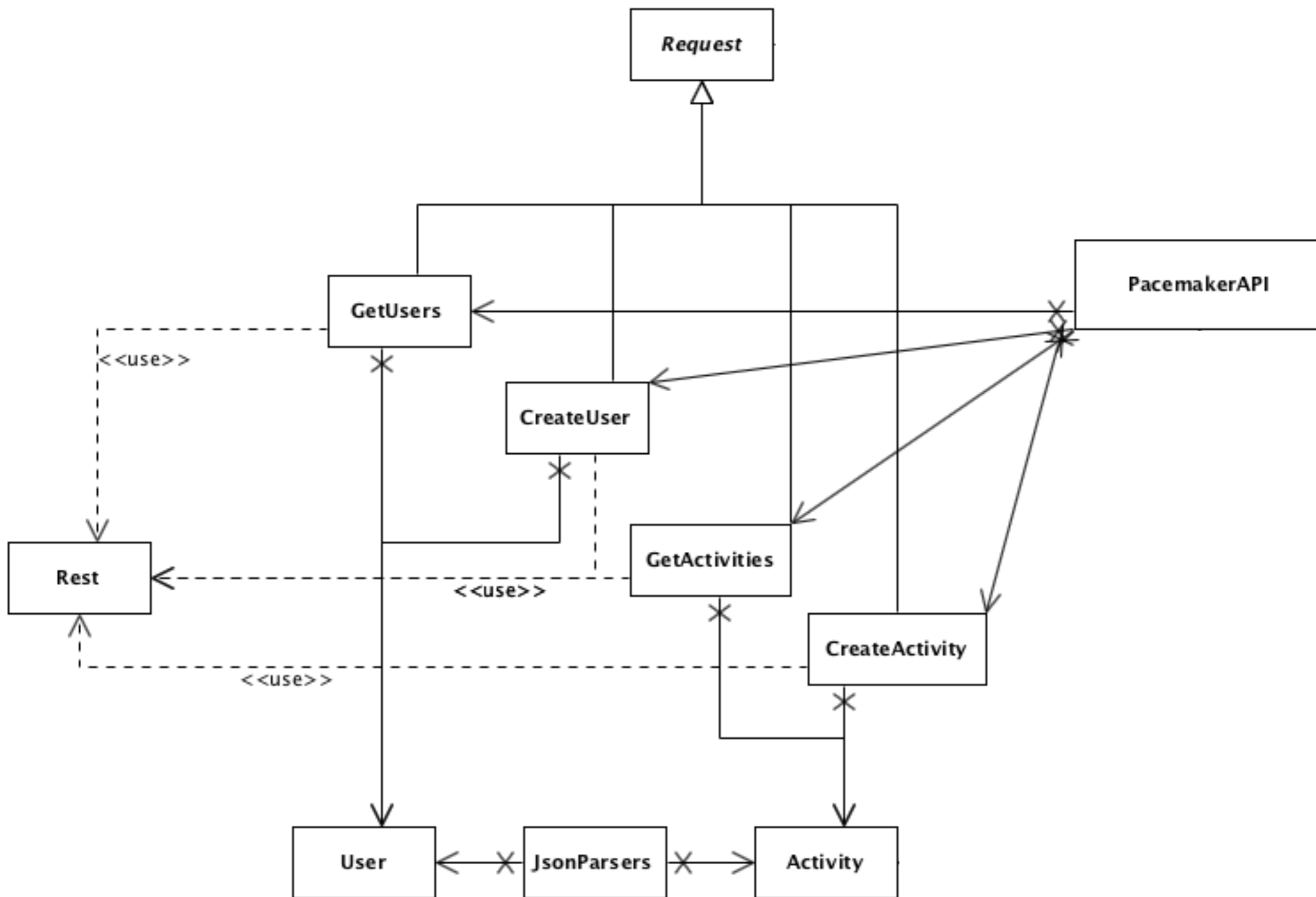
    public GetActivities(Context context, User user, Response<Activity> callback, String message)
    {
        super(context, callback, message);
        this.user = user;
    }

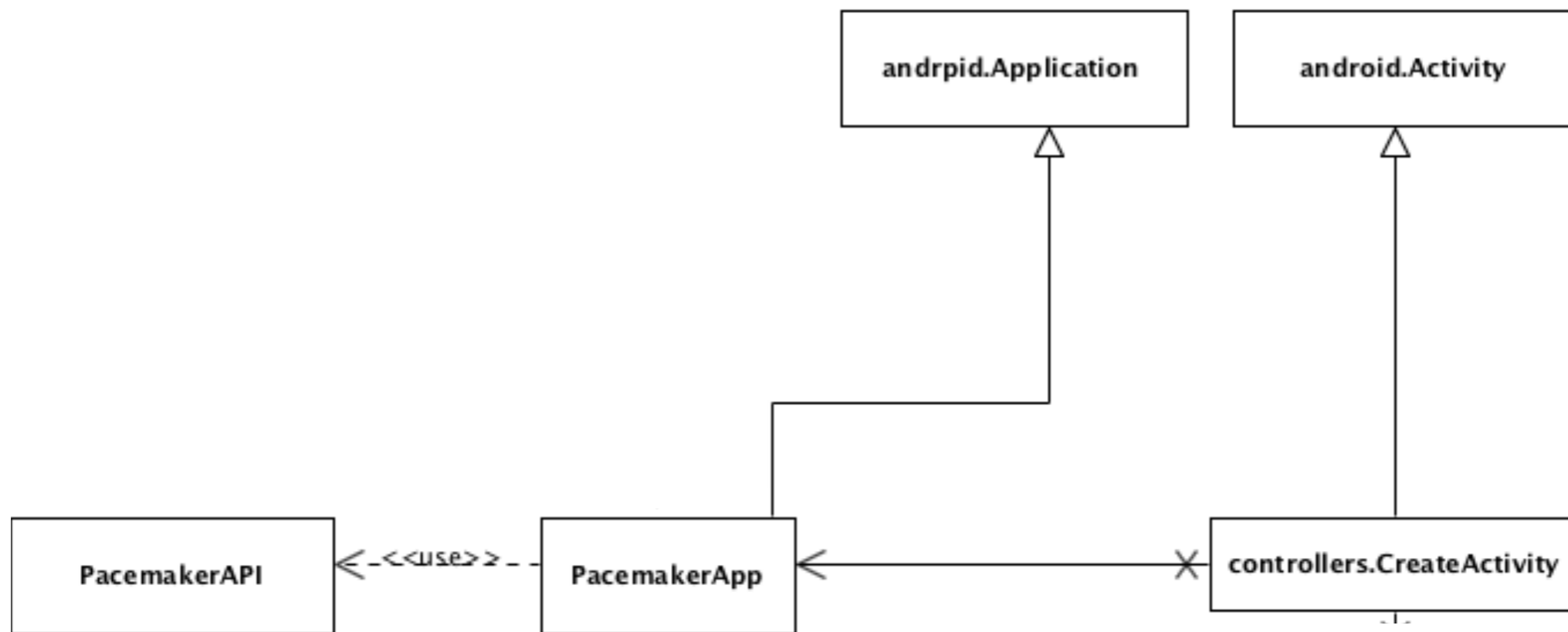
    @Override
    protected List<Activity> doRequest(Object... params) throws Exception
    {
        String response = Rest.get("/api/users/" + user.id + "/activities");
        List<Activity> ActivityList = JsonParser.json2Activities(response);
        return ActivityList;
    }
}

class CreateActivity extends Request
{
    private User user;

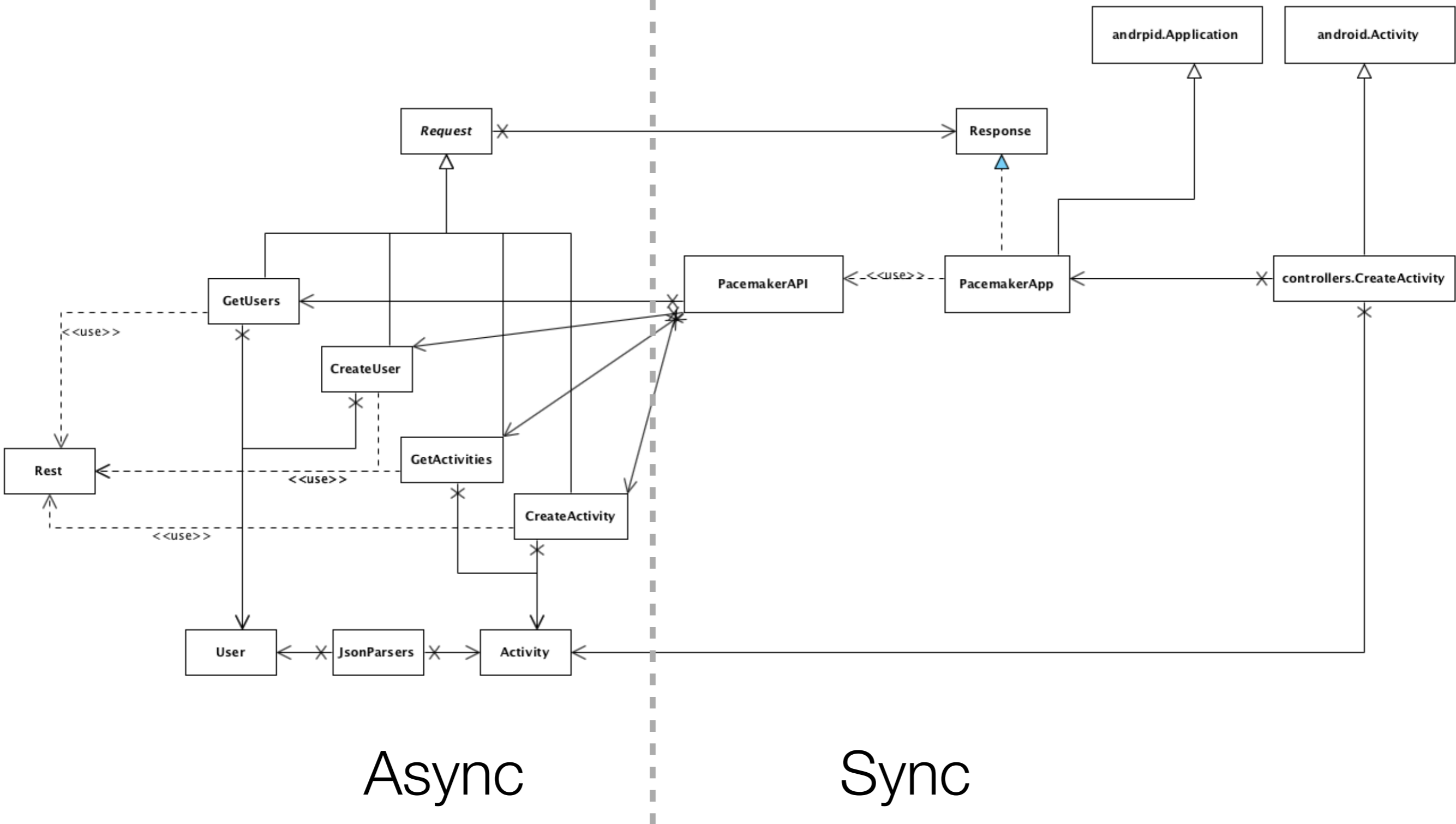
    public CreateActivity(Context context, User user, Response<Activity> callback, String message)
    {
        super(context, callback, message);
        this.user = user;
    }

    @Override
    protected Activity doRequest(Object... params) throws Exception
    {
        String response = Rest.post ("/api/users/" + user.id + "/activities", JsonParser.activity2Json(params[0]));
        return JsonParser.json2Activity(response);
    }
}
```





Half Sync / Half Async



PacemakerApp

- pacemaker is a Facade, encapsulating access to model
- It now encapsulates synchronisation behaviour,

```
public class PacemakerApp extends Application implements Response<User>
{
    private Map<String, User> users = new HashMap<String, User>();
    private User loggedInUser;

    public void connectToPacemakerAPI(Context context)
    {
        PacemakerAPI.getUsers(context, this, "Retrieving list of users");
    }
    @Override
    public void setResponse(List<User> aList)
    {
        connected = true;
        for (User user : aList)
        {
            users.put(user.email, user);
        }
    }
    @Override
    public void setResponse(User user)
    {
        connected = true;
        users.put(user.email, user);
        activities.put(user.email, new ArrayList<Activity>());
    }
    @Override
    public void errorOccurred(Exception e)
    {
        connected = false;
        Toast toast = Toast.makeText(this, "Failed to connect to Pacemaker Service", Toast.LENGTH_SHORT);
        toast.show();
    }
    public void registerUser(Context context, User user)
    {
        PacemakerAPI.createUser(context, this, "Registering new user", user);
    }
    public boolean loginUser(String email, String password)
    {
        loggedInUser = users.get(email);
        if (loggedInUser != null && !loggedInUser.password.equals(password))
        {
            loggedInUser = null;
        }
        return loggedInUser != null;
    }
    public void logout()
    {
        loggedInUser = null;
    }
}
```



Pacemaker

Login

Sign up

```
public class Welcome extends Activity
{
    PacemakerApp app;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_welcome);
        app = (PacemakerApp) getApplication();
        app.connectToPacemakerAPI(this);
    }

    public void loginPressed (View view)
    {
        startActivity (new Intent(this, Login.class));
    }

    public void signupPressed (View view)
    {
        startActivity (new Intent(this, Signup.class));
    }
}
```




Signup

Sign up for the Pacemaker

Enter details below

```
public class Signup extends Activity
{
    private PacemakerApp app;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_signup);
        app = (PacemakerApp) getApplication();
    }

    public void registerPressed (View view)
    {
        TextView firstName = (TextView) findViewById(R.id.firstName);
        TextView lastName = (TextView) findViewById(R.id.lastName);
        TextView email = (TextView) findViewById(R.id.Email);
        TextView password = (TextView) findViewById(R.id.Password);

        User user = new User (firstName.getText().toString(),
                               lastName.getText().toString(),
                               email.getText().toString(),
                               password.getText().toString());

        app.registerUser(this, user);
        startActivity (new Intent(this, Login.class));
    }
}
```

- no change from standalone version



Login to Donation

You must be registered


```
public class Login extends Activity
{
    PacemakerApp app;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
    }

    public void signInPressed (View view)
    {
        app = (PacemakerApp) getApplication();

        TextView email      = (TextView) findViewById(R.id.loginEmail);
        TextView password   = (TextView) findViewById(R.id.loginPassword);

        boolean loggedIn = app.loginUser(email.getText().toString(),
                                         password.getText().toString());

        if (loggedIn)
        {
            startActivity (new Intent(this, CreateActivity.class));
        }
        else
        {
            Toast toast = Toast.makeText(this, "Invalid Credentials",
                                         Toast.LENGTH_SHORT);

            toast.show();
        }
    }
}
```

- no change from standalone version

Enter Activity Details

Distance

20

0

1

```
public class CreateActivity extends android.app.Activity implements Response <Activity>
{
    private PacemakerApp app;
    //...

    public void createActivityButtonPressed (View view)
    {
        double distance = distancePicker.getValue();
        Activity activity = new Activity (activityType.getText().toString(),
                                         activityLocation.getText().toString(),
                                         distance)

        app.createActivity(activity);
    }

    @Override
    public void setResponse(List<Activity> aList)
    {}

    @Override
    public void setResponse(Activity anObject)
    {}

    @Override
    public void errorOccurred(Exception e)
    {
        Toast toast = Toast.makeText(this, "Failed to create Activity", Toast.LENGTH_SHORT);
        toast.show();
    }
}
```

 ActivitiesList

Activities

cycle

fenor

19.0

```
public class ActivitiesList extends android.app.Activity implements Response <Activity>
{
    private PacemakerApp    app;
    private ListView        activitiesListView;
    private ActivityAdapter activitiesAdapter;
    private List<Activity>  activities = new ArrayList<Activity>();

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activities_list);

        app = (PacemakerApp) getApplication();

        activitiesListView = (ListView) findViewById(R.id.activitiesListView);
        activitiesAdapter = new ActivityAdapter(this, activities);
        activitiesListView.setAdapter(activitiesAdapter);

        app.getActivities(this, this);
    }

    @Override
    public void setResponse(List<Activity> aList)
    {
        activitiesAdapter.activities = aList;
        activitiesAdapter.notifyDataSetChanged();
    }

    @Override
    public void setResponse(Activity anObject)
    {
    }

    @Override
    public void errorOccurred(Exception e)
    {
        Toast toast = Toast.makeText(this, "Error Retrieving Activities...", Toast.LENGTH_SHORT);
        toast.show();
    }
}
```



ActivitiesList

Activities

cycle	fenor	19.0
-------	-------	------

```
class ActivityAdapter extends ArrayAdapter<Activity>
{
    private Context context;
    public List<Activity> activities;

    public ActivityAdapter(Context context, List<Activity> activities)
    {
        super(context, R.layout.activity_row_layout, activities);
        this.context = context;
        this.activities = activities;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent)
    {
        LayoutInflater inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

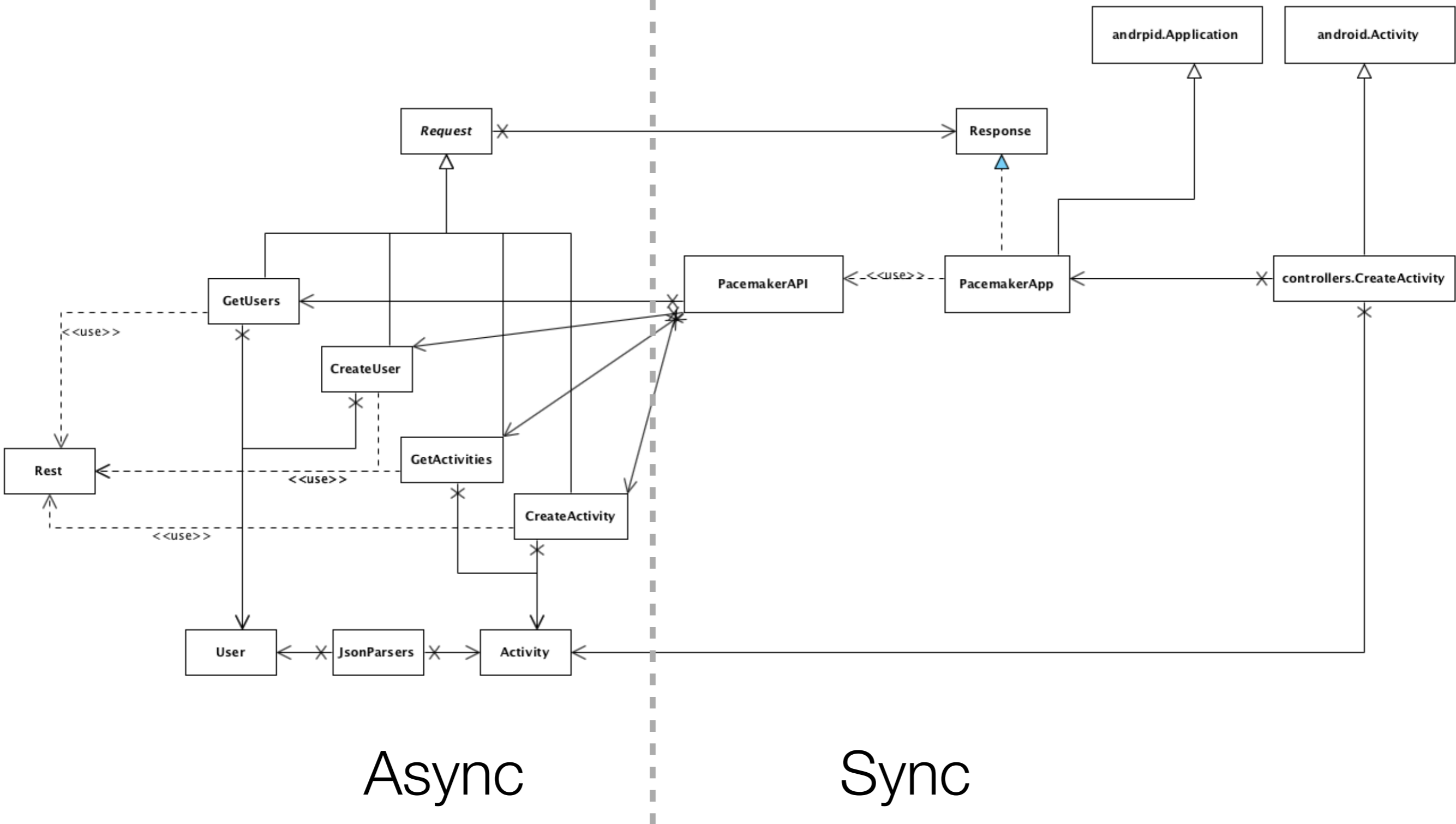
        View view = inflater.inflate(R.layout.activity_row_layout, parent, false);
        Activity activity = activities.get(position);
        TextView type = (TextView) view.findViewById(R.id.type);
        TextView location = (TextView) view.findViewById(R.id.location);
        TextView distance = (TextView) view.findViewById(R.id.distance);

        type.setText(activity.type);
        location.setText(activity.location);
        distance.setText(" " + activity.distance);
        return view;
    }

    @Override
    public int getCount()
    {
        return activities.size();
    }
}
```

- no change from standalone version

Half Sync / Half Async





Except where otherwise noted, this content is licensed under a Creative Commons Attribution-NonCommercial 3.0 License.

For more information, please see <http://creativecommons.org/licenses/by-nc/3.0/>

