

3. PERANCANGAN DAN IMPLEMENTASI

Dalam pembuatan aplikasi Manajemen Tugas ini penulis menggunakan bahasa pemrograman Java dan menggunakan Eclipse IDE sebagai perangkat lunak pendukung untuk mempermudah pembuatan aplikasi Android dan aplikasi pada Web Server. Dalam proses perancangan pembuatan Aplikasi Manajemen Tugas ini terdiri dari beberapa tahap.

3.1 Analisa Program

Pada Analisa Program penulis akan menjelaskan tentang gambaran umum dari aplikasi Manajemen Tugas ini. Aplikasi ini diperuntukkan bagi umum untuk mencatat tugas sehari-hari dan sebagai pengingat. Penggunaan aplikasi ini bisa digunakan kapan saja asalkan pengguna sudah mempunyai smartphone untuk menginstalnya kedalam sebuah mobile Android yang sesuai dengan versi yang dibuat pada aplikasi ini yaitu sistem operasi Android 2.2 dan juga dapat digunakan pada sistem operasi di atasnya.

Fitur utama dari aplikasi Manajemen Tugas ini adalah sebagai pengingat tugas di dalam kehidupan sehari-hari. Dan juga data bisa disimpan di cloud.

3.2 Perancangan Aplikasi

Dalam pembuatan aplikasi ini dibagi menjadi dua langkah utama, yang pertama membuat server yang akan menjadi API di cloud server dan sebagai penyimpanan data kemudian langkah berikutnya adalah pembuatan aplikasi di sisi client yaitu Aplikasi Android.

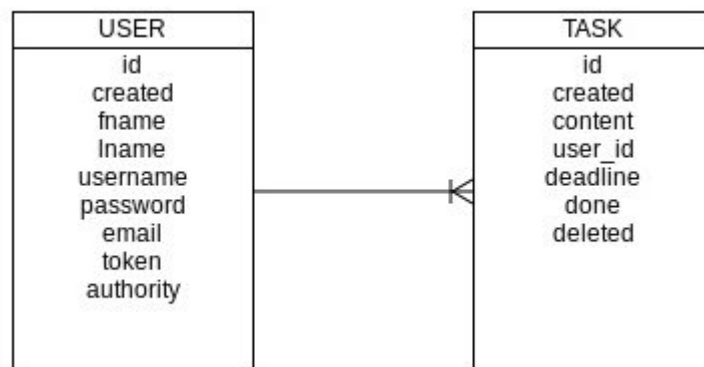
Di dalam pembuatan aplikasi server yang pertama penulis akan membuat rancangan database dan, ERD dan Struktur Tabel, kemudian di sisi client Aplikasi Android yang pertama di lakukan adalah membuat rancangan struktur navigasi dan rancangan tampilan (Mockup)

3.2.1 Aplikasi Server

Di dalam pembuatan aplikasi web service di server cloud ini penulis perlu membuat rancangan database ERD dan Struktur Tabel terlebih dahulu.

3.2.1.1 Rancangan Database

Sebelum membuat database dan tabel pada DBMS penulis akan menuliskan rancangan database secara konseptual melalui Entity Relationship Diagram (ERD). Pada penulisan ilmiah ini penulis akan membuat rancangan Entity Relationship Diagram (ERD) dimana terdapat 2 buah tabel master . Berikut ini adalah gambar rancangan ERD:



Pada gambar ERD diatas terdapat 2 buah tabel master user dan task, dan dengan relasi satu user dapat mempunyai beberapa task (one to many)

Setelah relasi antar tabel dibuat, maka langkah selanjutnya adalah mentransformasikan relasi tersebut pada database. Berikut ini adalah struktur tabel yang akan dibuat pada database.

Nama Field	Tipe Data	Keterangan
id	Varchar(50)	Primary Key
created	Timestamp	
fname	Varchar(25)	
lname	Varchar(25)	
username	Varchar(25)	
password	Varchar(25)	
email	Varchar(25)	
token	Varchar(50)	
authority	Varchar(10)	

Nama Field	Tipe Data	Keterangan
id	Varchar(50)	Primary Key
created	Timestamp	
content	Varchar(255)	
user_id	Varchar(50)	Foreign Key
deadline	Timestamp	
done	Integer	
deleted	Integer	

3.2.1.2 Membuat Database dan Tabel pada PostgreSQL

Membuat database dan tabel pada PostgreSQL bisa dilakukan dengan dua cara, yakni melalui console dengan mengetikkan perintah – perintah SQL secara manual atau bisa juga melalui pgAdmin. Pada penulisan ilmiah ini penulis akan membuat database dan tabel menggunakan console.

```
CREATE TABLE etask_user (
  id character varying(255) NOT NULL,
  created timestamp without time zone,
  email character varying(255),
  first_name character varying(255),
  last_name character varying(255),
  password character varying(255),
  token character varying(255),
  username character varying(255),
  authority character varying(255),
```

```

        CONSTRAINT etask_user_pkey PRIMARY KEY (id),
        CONSTRAINT etask_user_email_key UNIQUE (email),
        CONSTRAINT etask_user_username_key UNIQUE (username)
    )
    WITH (
        OIDS=FALSE
    );

CREATE TABLE task (
    id character varying(255) NOT NULL,
    created timestamp without time zone,
    content character varying(255),
    deadline timestamp without time zone,
    deleted boolean,
    done boolean,
    user_id character varying(255),
    CONSTRAINT task_pkey PRIMARY KEY (id),
    CONSTRAINT fk3635856d590aaa FOREIGN KEY (user_id)
    REFERENCES etask_user (id) MATCH Unknown
    ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
    OIDS=FALSE
);

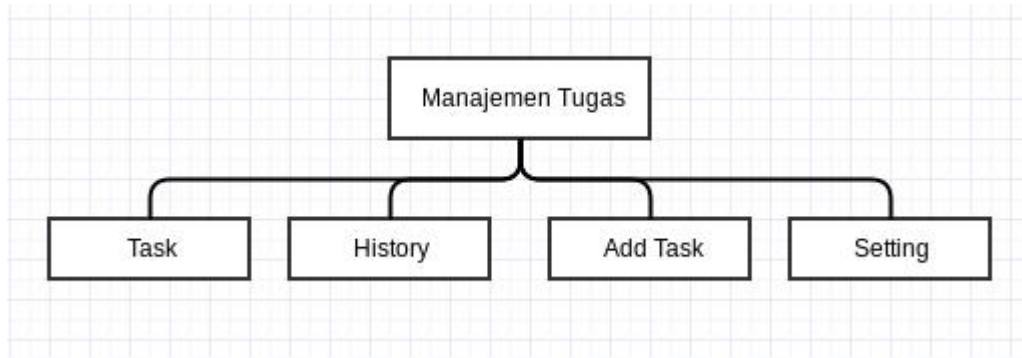
```

3.2.2 Aplikasi Android

Dalam pembuatan aplikasi Android yang berfungsi sebagai client ini terlebih dahulu penulis membuat rancangan struktur navigasi kemudian dilakukan rancangan tampilan aplikasi utama dan pembuatan rancangan tampilan pengaturan.

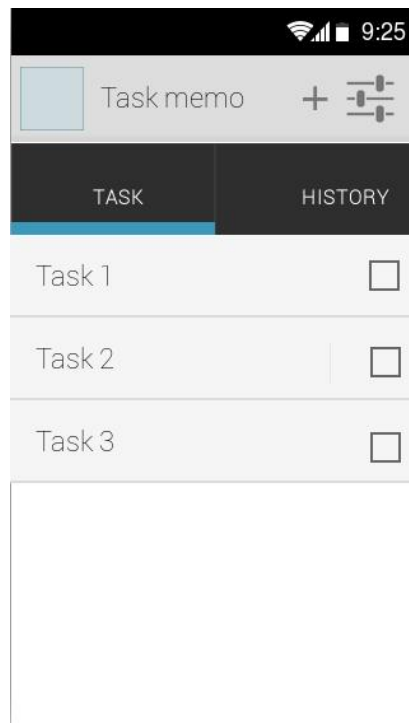
3.2.2.1 Stuktur Navigasi

Dalam proses perancangan aplikasi, struktur navigasi sangat penting. Struktur navigasi dapat menjelaskan hubungan antar halaman dan juga dapat menjelaskan mengenai alur cerita sebuah program atau aplikasi, juga memberi kemudahan menganalisa keteraktifan seluruh obyek dan bagaimana pengaruh keinteraktifannya terhadap pengguna.

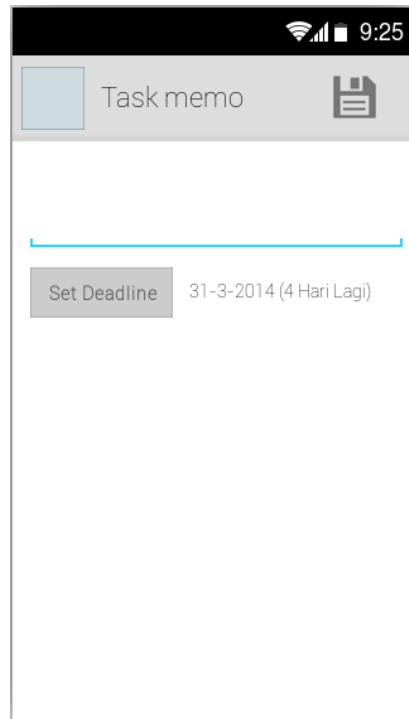


3.2.2.1 Rancangan Tampilan

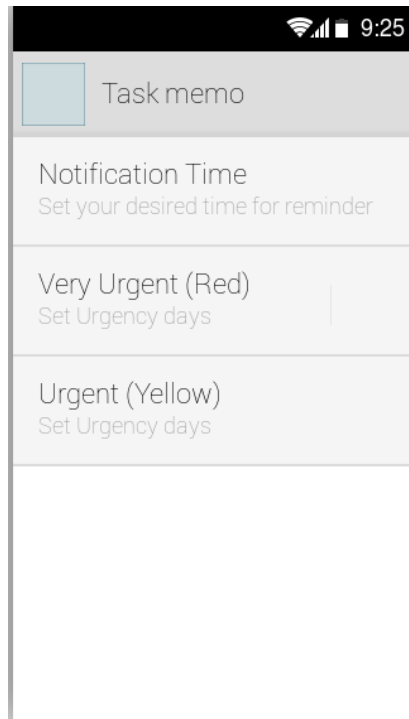
Rancangan tampilan aplikasi ini terdiri dari rancangan halaman aplikasi utama yang berupa daftar task atau tugas di halaman ini akan menampilkan task atau tugas yang pernah penulis isi dan informasi berapa hari task ini harus selesai (deadline). Kemudian di bagian action bar terdapat dua menu yaitu menu add untuk menambah task dan menu setting untuk pengaturan aplikasi.



Untuk tampilan pada halaman add terdapat form isian untuk informasi task di antaranya adalah content dan tanggal deadline. Deadline ini di gunakan untuk penghitungan berapa hari task ini harus selesai yang nantinya di tampilkan pada halaman aplikasi utama.



Kemudian di tampilan setting atau pengaturan terdapat beberapa pilihan pengaturan diantaranya, Notification Time yang di fungsinya untuk mengatur pada jam berapa notifikasi akan di tampilkan. Notifikasi ini berisi tentang informasi deadline yang terdekat, kemudian ada pengaturan urgency, di pengaturan ini ada dua kategori pertama kategori Very Urgent dan Urgent di sini penulis bisa mengatur berapa hari sebelum deadline di sebut Very Urgent atau Urgent yang nanti akan di tampilkan warna berbeda di halaman aplikasi utama



3.3 Pembuatan Aplikasi

Setelah melalui tahap perancangan aplikasi kemudian lakukan langkah-langkah pembuatan aplikasi sesuai dengan rancangan sistem dan rancangan tampilan yang telah dibuat sebelumnya. Karena sistem yang akan penulis buat ada dua komponen yaitu Client dan Server maka penulis akan membuat di sisi server terlebih dahulu. Tapi sebelum itu penulis akan melakukan instalasi Eclipse dan Plugin-plugin yang di perlukan untuk membuat aplikasi ini.

3.3.1 Instalasi IDE

Langkah-langkah pembuatan aplikasi Aplikasi Manajemen Tugas. Pertama-tama diawali dengan meng-instal jdk , Proses peng-*instal*-an ini dilakukan pada sistem operasi Centos, jika *software* ini belum ada pada komputer yang digunakan untuk meng-compile skrip *coding* yang dibuat.

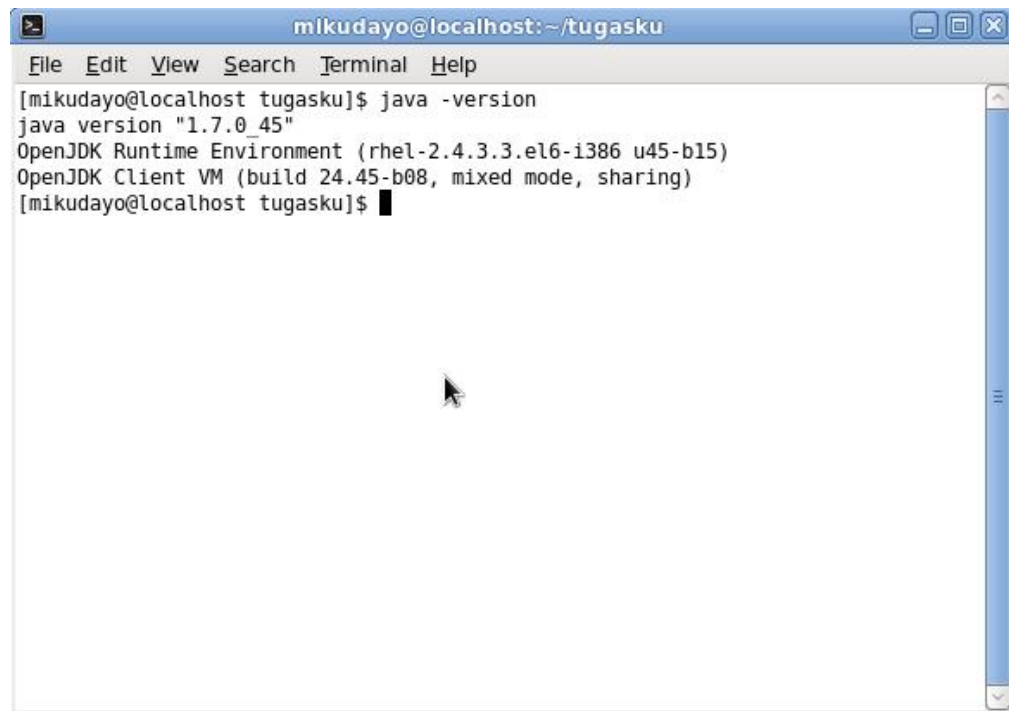
Software jdk bisa didapatkan dengan cara *men-download* atau membeli cd aslinya. Namun pada kesempatan ini penulis *men-download* website www.oracle.com/technetwork/java/javase/downloads/index.html. Untuk instalasinya penulis cukup mengeksekusi file binary yang sebelumnya penulis download dengan perintah di konsol

```
rpm -i jdk-7u25-linux-i586.rpm
```

Selanjutnya ketika instalasi selesai penulis bisa cek apakah java sudah terinstall dengan baik di komputer penulis dengan mengeksekusi perintah

```
java -version
```

Jika jdk sudah terinstall dengan baik maka yang akan muncul di konsol adalah seperti di bawah ini



```
mikudayo@localhost:~/tugasku
File Edit View Search Terminal Help
[mikudayo@localhost tugasku]$ java -version
java version "1.7.0_45"
OpenJDK Runtime Environment (rhel-2.4.3.3.el6-i386 u45-b15)
OpenJDK Client VM (build 24.45-b08, mixed mode, sharing)
[mikudayo@localhost tugasku]$
```

Kemudian selanjutnya penulis akan menginstall Eclipse IDE, untuk mendapatkannya penulis bisa mendownload dari website <https://www.eclipse.org/downloads/> dan pilih untuk versi yang **Eclipse IDE for Java EE Developers** karena di versi ini merupakan versi terlengkap yang di dalamnya sudah terdapat plugin-plugin yang akan penulis gunakan. Setelah berhasil download, instalasi eclipse ini cukup mudah yaitu tunggal ekstrak di tempat yang di inginkan. Untuk menjalankan eclipse ini penulis tinggal mengeksekusi binary yang bernama eclipse di dalam archive yang penulis download sebelumnya.

Karena eclipse yang penulis download ini belum terdapat Plugin untuk membuat aplikasi android maka penulis perlu menginstall secara manual.

1. Di Eclipse pilih menu Help > Install New Software
2. Klik Add di pojok kanan atas
3. Di dalam dialog Add Repository masukkan url <https://dl-ssl.google.com/android/eclipse/> di bagian Location.
4. Klik OK
5. Di dialog Available Software Dialog, pilih checkbox di sebelah Developer Tools kemudian klik Next
6. Di dialog selanjutnya lanjutkan dengan klik tombol Next
7. Kemudian baca dan centang License Agreements kemudian klik Finish
8. Setelah instalasi selesai restart Eclipse

kemudian yang di perlukan selanjutnya adalah instalasi sdk android di eclipse

Pertama penulis perlu mendownload Android SDK dari website resmi <http://developer.android.com/sdk/index.html> dan memilih versi yang [android-sdk_r22.3-linux.tgz](#) Kemudian ekstrak di tempat yang di inginkan. Kemudian buka eclipse dan lakukan langkah-langkah berikut

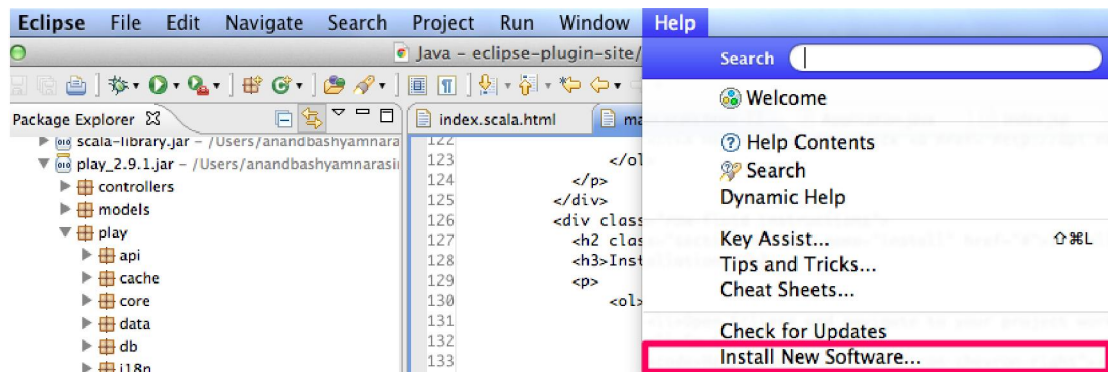
1. Di Eclipse pilih menu Window > Preferences

2. Di sebelah navigasi kiri pilih Android
3. Kemudian masukkan lokasi sdk yang sudah penulis ekstrak di bagian SDK Location, Kemudian klik Apply dan OK

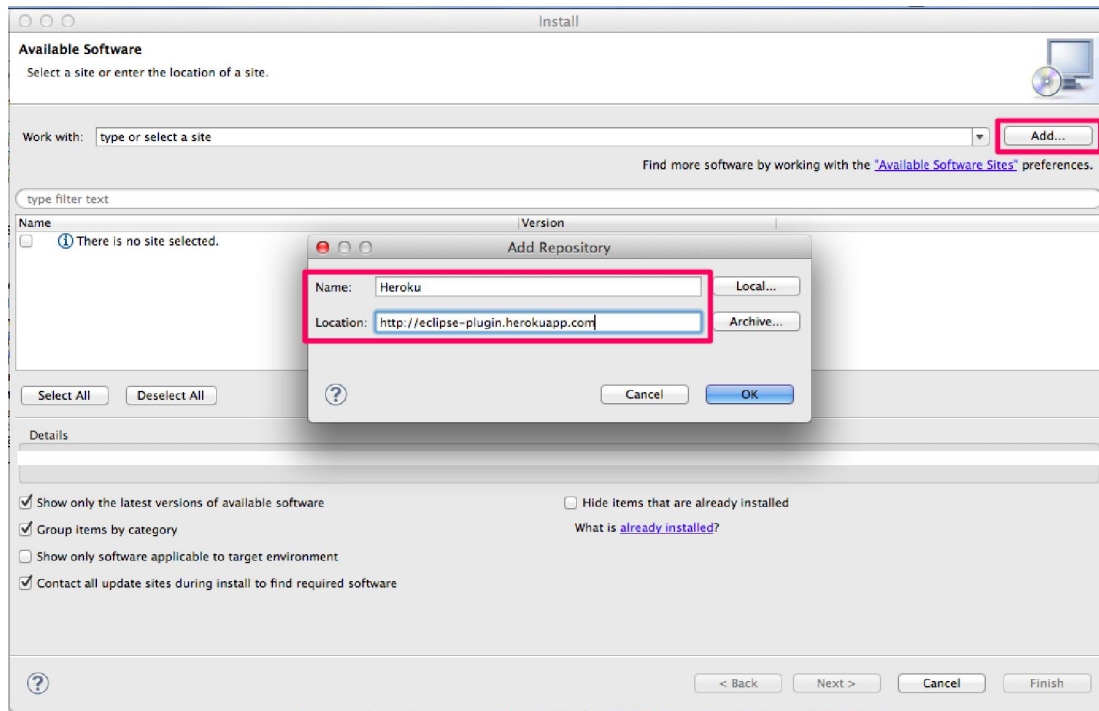
Setelah selesai untuk instalasi yang di butuhkan untuk membuat aplikasi Android selanjutnya penulis akan menginstall plugin untuk membuat aplikasi yang dapat di publish ke cloud provider yaitu Heroku

Langkah pertama penulis registrasi akun baru di website heroku <https://id.heroku.com/signup/devcenter> setelah mendapatkan akun penulis lanjutkan untuk instalasi di eclipse

1. Di tampilan utama Eclipse pilih menu Help > Install New Software



2. Klik Add
3. Masukkan url dengan nama Heroku dan <https://eclipse-plugin.herokuapp.com/install> di bagian location kemudian klik OK



4. Pilih Checkbox di sebelah kiri “Heroku Eclipse Integration”
5. Klik Next dua kali
6. Accept License Agreements dan kemudian Finish
7. Restart Eclipse

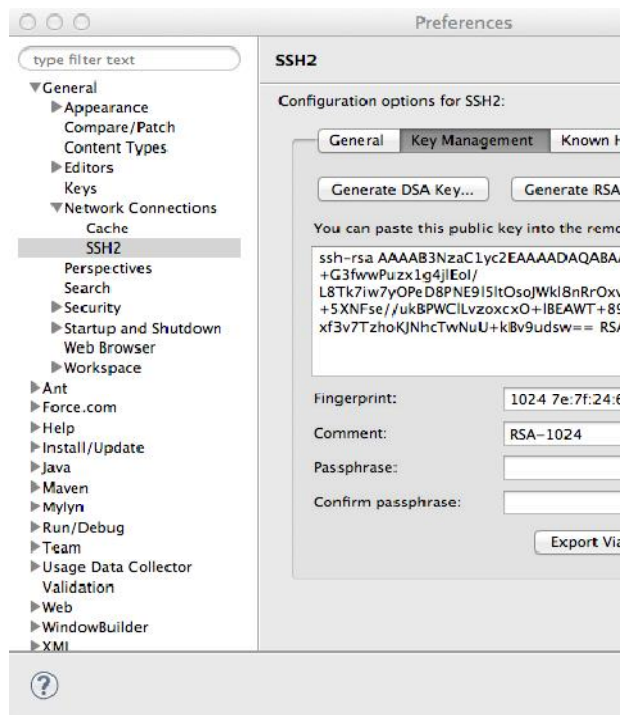
Setelah instalasi plugin heroku selesai selanjutnya penulis akan mengkonfigurasi API Key untuk heroku agar eclipse penulis terintegrasi dengan Heroku.

1. Di tampilan utama Eclipse pilih Eclipse > Preferences
2. Pilih Menu Herku di bagian kiri
3. Masukkan email dan password yang sebelumnya sudah di registrasi
4. Klik Login, Jika login sukses maka API Key akan otomatis terisi.

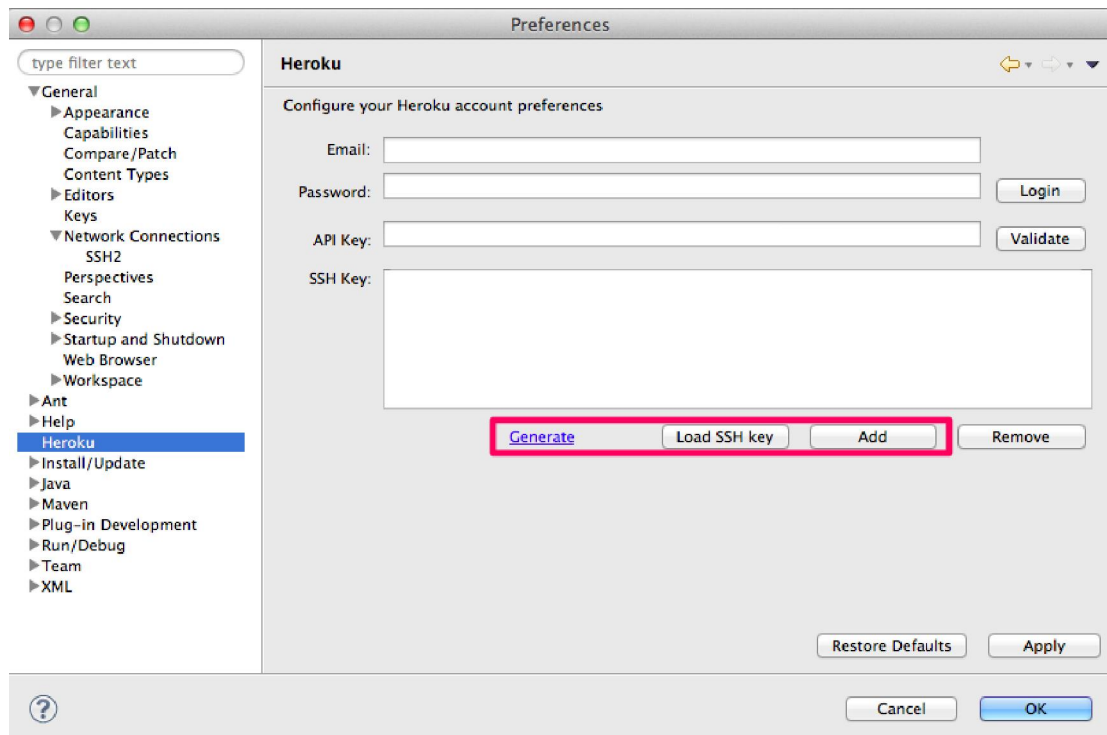
Membuat SSH Key

Untuk men-deploy aplikasi penulis ke Heroku, plugin eclipse Heroku ini menggunakan Git. Jadi selanjutnya penulis akan menginstall plugin Git di Eclipse, langkah-langkah untuk instalasi ini hampir sama seperti sebelumnya penulis menginstall plugin heroku yang membedakan adalah url plugin untuk git ini adalah <http://download.eclipse.org/egit/updates>. Sebuah SSH Key di butuhkan untuk untuk komunikasi git ke server heroku. Penulis membutuhkan beberapa konfigurasi untuk pertama kali membuat aplikasi di Heroku. Pertama yang perlu penulis lakukan adalah meng-generate SSH Key baru. Berikut adalah step step untuk mendapatkannya

1. Di Eclipse pilih menu Eclipse > Preferences > General Network Connections > SSH2
2. Pilih tab Key Managemen
3. Klik Generate SSH Key



4. Klik Save Private Key, kemudian Klik OK
5. Di Eclipse pilih menu Eclipse > Preference > Heroku
6. Di dalam kolom SSH Key klik generate



7. Klik Add dan OK.

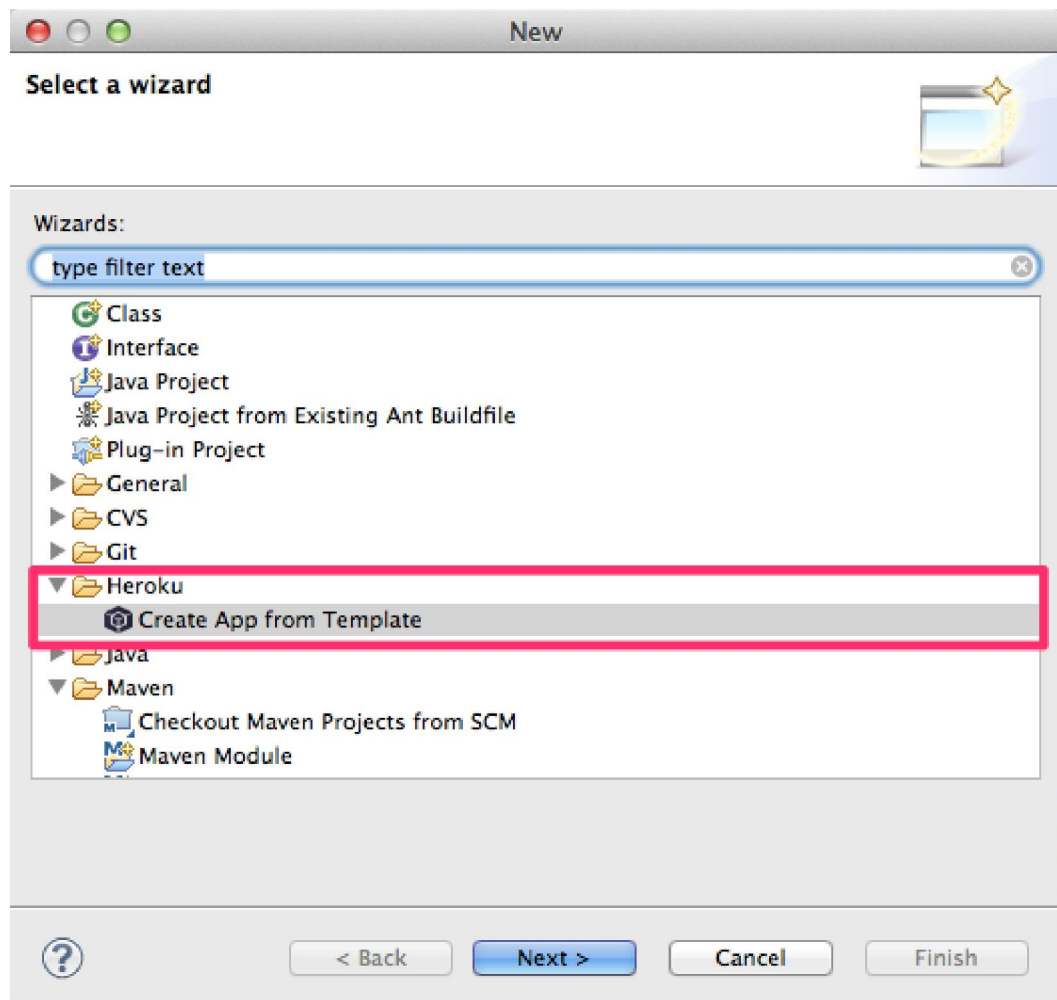
3.3.2 Aplikasi Server

Setelah selesai dengan instalasi IDE dan plugin-plugin langkah selanjutnya adalah membuat aplikasi server dimana di tahap ini pertama penulis harus membuat sebuah database dan table di Postgresql dengan schema yang sudah penulis buat sebelumnya.

3.3.1.1 Membuat Sring MVC Project

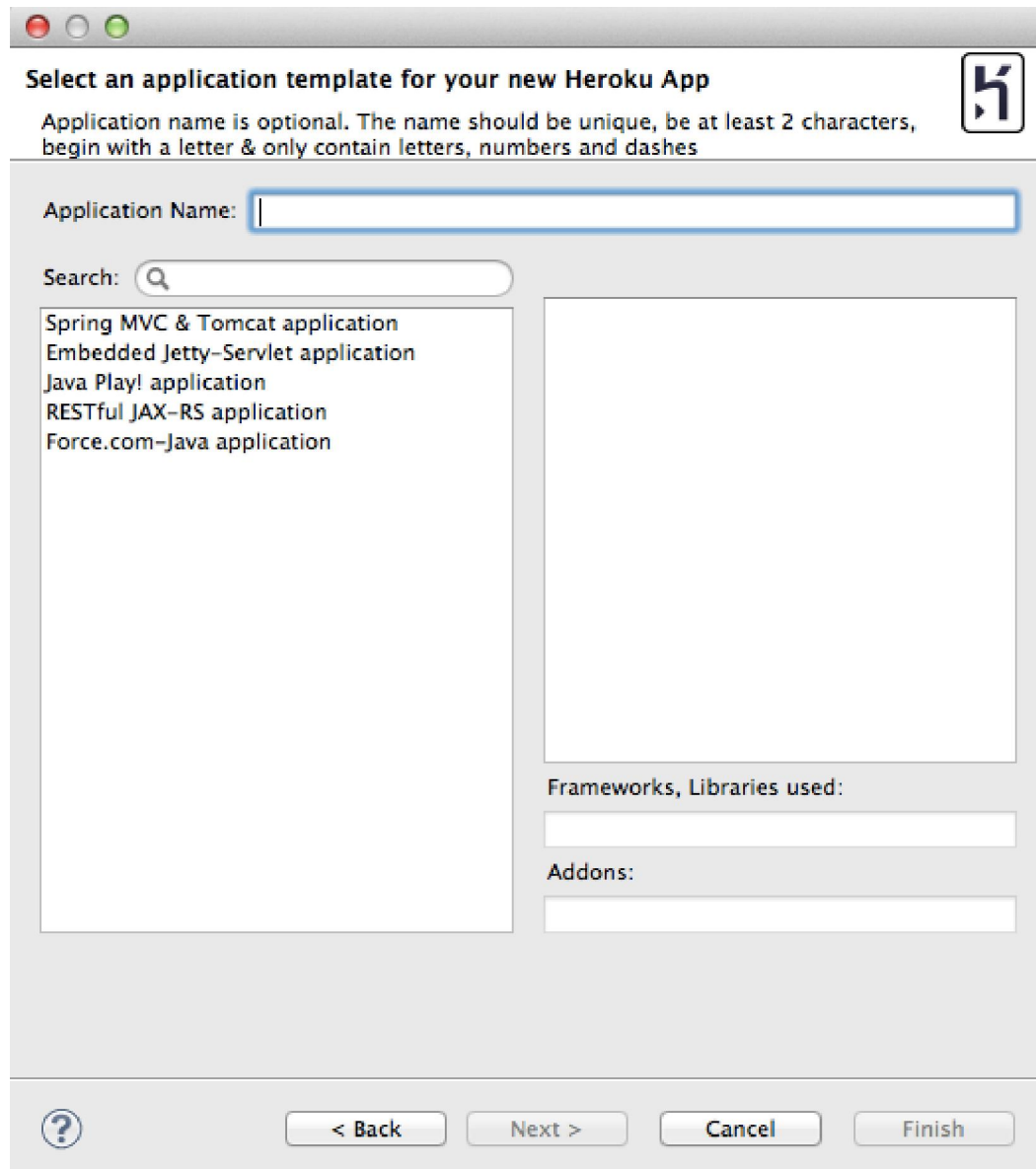
Untuk pertama kali membuat project penulis akan menggunakan template project yang sudah di sediakan oleh Heroku Plugin Eclipse yang sudah penulis install sebelumnya.

1. Buka menu File > New > Other kemudian buka bagian Heroku



2. Pilih Create App from Template kemudian klik Next
3. Di dalam nama masukkan TugasKu

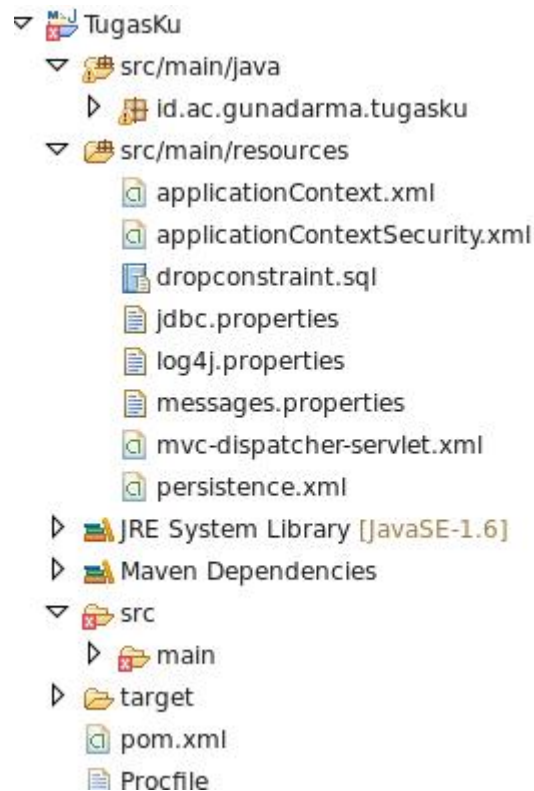
4. Kemudian di dalam pilihan template pilih Spring MVC & Tomcat Application



The screenshot shows a window titled "Select an application template for your new Heroku App" with the Heroku logo in the top right corner. Below the title, a note states: "Application name is optional. The name should be unique, be at least 2 characters, begin with a letter & only contain letters, numbers and dashes". There is a text input field for "Application Name:". Below this is a "Search:" field with a magnifying glass icon. A list of application templates is displayed in a scrollable area: "Spring MVC & Tomcat application", "Embedded Jetty-Servlet application", "Java Play! application", "RESTful JAX-RS application", and "Force.com-Java application". To the right of the list is a large empty rectangular box. Below the list and box, there are two input fields labeled "Frameworks, Libraries used:" and "Addons:". At the bottom of the window, there is a help icon (question mark in a circle) and four buttons: "< Back", "Next >", "Cancel", and "Finish".

5. Klik Finish untuk membuat aplikasi dari template yang sudah ada.

Ketika selesai maka akan ada project dengan struktur seperti gambar di bawah ini.



3.3.1.2 Implementasi Data Access Object (DAO) Pattern

Setelah rancangan konseptual sistem selesai dibuat. Langkah selanjutnya adalah mengimplementasikan rancangan tersebut pada bahasa pemrograman. Bahasa pemrograman yang digunakan adalah Java. Dimana proses peng codingan program menggunakan teknik Data Access Object (DAO) Pattern yang merupakan salah satu design pattern pada arsitektur Java EE. Ada beberapa tahapan dalam mengimplementasikan Data Access Object Pattern. Tahapan – tahapan tersebut adalah :

1. Membuat class untuk setiap entitas
2. Membuat DAO Interface
3. Membuat class implementasi DAO

3.3.1.3 Membuat class untuk setiap entitas

Entity class merupakan class yang merepresentasikan entitas di dalam database. Class ini berfungsi sebagai setter dan getter di dalam program. Setiap class terdiri dari method – method yang merepresentasikan field di dalam entitas tersebut.

Adapun pada penulisan ilmiah ini penulis hanya menuliskan script untuk class entitas Task, sedangkan untuk class entitas lainnya bisa dilihat pada lampiran kode program. Berikut adalah script untuk class entitas Task:

```
@Entity
@Table(name = "task")
public class Task extends AbstractEntity {

    private static final long serialVersionUID = -
6234938765346643021L;
    private String content;
    private User user;
    private Date deadline;
    private boolean done;
    private boolean deleted = false;

    @Column
    public String getContent() {
        return content;
    }

    public void setContent(String content) {
        this.content = content;
    }

    @ManyToOne(cascade = CascadeType.ALL)
    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    @Column
    public Date getDeadline() {
        return deadline;
    }

    public void setDeadline(Date deadline) {
```

```

        this.deadline = deadline;
    }

    @Column
    public boolean isDone() {
        return done;
    }

    public void setDone(boolean done) {
        this.done = done;
    }

    @Override
    public String toString() {
        return "Task [content=" + content + ", user=" + user + ",
deadline="
+ deadline.toString() + ", done=" + done + "]";
    }

    @Column
    public boolean isDeleted() {
        return deleted;
    }

    public void setDeleted(boolean deleted) {
        this.deleted = deleted;
    }
}

```

Entitas Task yang terdapat pada database memiliki 5 field, begitu juga pada saat pembuatan entity class. Setiap class berisi method – method yang sesuai dengan jumlah field. Satu buah field memiliki dua buah method, satu method sebagai setter dan satu method sebagai getter.

3.3.1.4 Membuat Implementasi DAO dan Interface

Aplikasi yang penulis buat hanya memiliki logika bisnis berupa melakukan operasi dasar CRUD (Create, Read, Update, Delete) pada basis data jadi service yang penulis miliki seakan-akan hanya merupakan kepanjangan dari DAO tier yang

telah penulis buat sebelumnya karena memang DAO tier yang berurusan langsung dengan basis data bukan Service tier. Seperti pada DAO tier penulis juga membuat sebuah interface dan sebuah kelas implementasi.

Di bawah ini adalah salah satu contoh DAO Service dari object User

```
package id.ac.gunadarma.tugasku.security;

import id.ac.gunadarma.tugasku.model.User;
import id.ac.gunadarma.tugasku.model.Role;
import id.ac.gunadarma.tugasku.model.base.EntityListWrapper;

import java.util.List;

public interface UserService {

    public void removeUser(Integer id);

    public List<User> listUser();

    public void addUser(User user);

    public void updateUser(User user);

    public User getUserByID(Integer id);

    public User getUser(String username, String password);

    public User findByName(String username);

    public User updateToken(User user, String generatedToken);

    public User getUserByToken(String accessToken);

    public Role getRoleById(String id);

    public Role getRoleByAuth(String auth);

    public EntityListWrapper<User> all(int max, int page);

}
```

Di bawah ini adalah salah satu contoh Implementasi dari DAO Service User yang sudah penulis buat di atas.

```

@Repository
@Transactional(readOnly = true)
public class UserServiceImpl implements UserService {
    @PersistenceContext
    private EntityManager entityManager;

    @Transactional(readOnly=false)
    public void addUser(User user) {
        try {
            entityManager.persist(user);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public User getUserByID(Integer id) {
        Query query = entityManager.createQuery("SELECT u FROM
User u WHERE id = :id", User.class);
        query.setParameter("id", id);
        try {
            return (User) query.getSingleResult();
        } catch (NoResultException e) {
            return null;
        }
    }
}

```

3.3.1.5 Membuat API untuk security dan login

Karena aplikasi ini mempunyai fitur multi user dan dapat di akses oleh beberapa client seperti android, maka di sisi server penulis membutuhkan sebuah security handling yang menangani user login. Di dalam pemrograman web biasanya penulis hanya perlu membuat username dan password untuk mengakses data, tetapi dengan metode REST API Client - Server ini penulis menggunakan Token untuk Autorisasi.

Jadi Langkah pertama yang akan penulis lakukan adalah membuat Controller yang menangani Login user dengan membawa token.

```

@RequestMapping(value = "/login/token", method =
RequestMethod.POST)
public @ResponseBody ResponseEntity<String> generateToken(

```

```

        @RequestParam("username") String username,
        @RequestParam("password") String password)
        throws JsonGenerationException, JsonMappingException,
        IOException {
            Map<String, String> map = new HashMap<String, String>();
            User user = null;
            try {
                user = userService.getUser(username, password);
                if (user.getToken() == null) {
                    String generatedToken =
                        UUID.randomUUID().toString().replace("-", "");
                    LOG.info("Generated security token: " +
                        generatedToken);
                    user = userService.updateToken(user, generatedToken);
                }
                LOG.info("Generated security token already exist: " +
                    user.getToken());
                map.put("statusCode", "00");
                map.put("statusMessage", "Success");
                map.put("secretToken", user.getToken());
            } catch (NoResultException e) {
                map.put("statusCode", "01");
                map.put("statusMessage", "Bad credentials");
                return new
                ResponseEntity<String>(mapper.writeValueAsString(map),
                HttpStatus.UNAUTHORIZED);
            }
            return new
            ResponseEntity<String>(mapper.writeValueAsString(map),
            HttpStatus.OK);
        }
    }

```

Di dalam kode di atas penulis menerima parameter yang di kirim oleh client username dan password kemudian dengan menggunakan dao service yang telah penulis buat penulis melakukan validasi apakah username dan password yang di kirim valid, kalau valid maka controller di atas akan menggenerate random token yang akan di kembalikan ke client. Kemudian dengan token ini aplikasi client bisa mengakses data yang ada di server.

Langkah selanjutnya adalah membuat security untuk melakukan validasi token yang di bawa oleh client untuk melakukan request data.

```

@Override
public boolean preHandle(HttpServletRequest request,
    HttpServletResponse response, Object handler) throws

```

```

Exception {
    String accessToken = request.getHeader("Authorization");
    User user = null;
    try {
        user = userService.getUserByToken(accessToken);
    } catch (NoResultException e) {
        e.printStackTrace();
    }
    RequestContextHolder.currentRequestAttributes().setAttribute(
"user", user, RequestAttributes.SCOPE_SESSION);
    if (request.getRequestURI().startsWith("/api/") && user ==
null) {

        response.sendError(javax.servlet.http.HttpServletResponse.SC_
UNAUTHORIZED);
        return false;
    }
    return true;
}

```

Di dalam kode di atas penulis melakukan validasi ketika client mengakses API penulis dengan prefix url api apakah token yang di bawa setiap request adalah valid, ketika valid maka request akan di lanjutkan ke controller.

3.3.1.6 Membuat API Controller untuk Manajemen Tugas

Selanjutnya penulis akan membuat sebuah Controller dimana semua request yang di butuhkan oleh client di proses. Berikut contoh salah satu controller untuk menambahkan task.

```

@RequestMapping(value = "/api/task/add", method =
RequestMethod.POST)
public @ResponseBody ResponseEntity<String> addTask(@ModelAttribute
Task task) throws JsonGenerationException, JsonMappingException,
IOException {
    Map<String, String> map = new HashMap<String, String>();
    try {
        task.setUser(TokenInterceptor.currentUser());
        LOG.info("Persisting Task [" + task.getId() + "]
" + task.toString());
        if (task.getId() != null) {
            taskService.add(task);
            Task t = taskService.get(task.getId());

```

```

        if (!"".equals(task.getContent()))
            t.setContent(task.getContent());
        if (task.getDeadline() != null)
            t.setDeadline(task.getDeadline());
        taskService.add(t);
    }else{
        task.setId(null);
        taskService.add(task);
    }
    map.put("statusCode", "00");
    map.put("statusMessage", "Success");
    map.put("id", task.getId());
} catch (Throwable e) {
    map.put("statusCode", "02");
    map.put("statusMessage", "Persist Error");
    map.put("id", "");
    LOG.error("E", e);
}
return new
ResponseEntity<String>(mapper.writeValueAsString(map),
HttpStatus.OK);
}

```

Di dalam kode di atas client akan mengakses API dengan method POST dan membawa parameter yang ada di object Task, kemudian melakukan penyimpanan ke database dengan menggunakan DAO Service yang telah penulis buat. Setelah penyimpanan data sukses penulis akan mengembalikan response sukses berupa Json.

3.3.3 Aplikasi Android

Setelah melalui tahap perancangan aplikasi, instalasi Eclipse dan membuat API server kemudian lakukan langkah-langkah pembuatan aplikasi sesuai dengan rancangan sistem dan rancangan tampilan yang telah dibuat sebelumnya.

Pembuatan aplikasi android ini dimulai dengan membuat Activity form utama di form ini yang nanti akan menampilkan list task dan history kemudian terdapat juga menu add task dan setting. Untuk pembuatan program ini pertama merancang tampilan (*layout*) dan audio yang berada di *folder resource* (res), program dan disimpan dalam bentuk (.xml) dan kemudian dilanjutkan membuat program (.java)

yang terdapat di dalam *folder source* (src) package aplikasi tersebut.

3.3.3.1 Pembuatan Form Utama

Tampilan awal Dari aplikasi ini yaitu *MainActivity* yang menampilkan list task dan history, pertama penulis buat rancangan tampilan nya di *layout XML*. Kemudian penulis panggil rancangan yang telah penulis buat di XML ke dalam program java. Berikut listing program *activity_main.xml*

```
<android.support.v4.view.ViewPager
xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/pager"
android:layout_width="match_parent"
android:layout_height="match_parent" />
```

Di *activity_main.xml* ini penulis menggunakan widget *ViewPager* untuk menampilkan 2 tab Task & History. Kemudian berikut untuk kode java yang terhubung dengan *activity_main.xml* di atas

```
final ActionBar actionBar = getActionBar();
actionBar.setHomeButtonEnabled(false);
actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
```

Inisialisasi object *ActionBar* di atas adalah untuk mengambil object dari *actionbar android* yang kemudian penulis konfigurasi sesuai kebutuhan, disini penulis akan mendisable home button dan mengatur mode navigasi dengan Tab. Selanjutnya untuk menambahkan Tab penulis akan menggunakan kode di bawah ini

```
for (int i = 0; i < mAppSectionsPagerAdapter.getCount(); i++) {
    actionBar.addTab(actionBar.newTab().setText(
        mAppSectionsPagerAdapter.getPageTitle(i)).setTabListener(this
    ));
}
```

Di perulangan di atas penulis akan mengulang 2x karena penulis telah menset jumlah tab ada 2 di method

```
@Override
public int getCount () {
    return 2;
}
```

Untuk penganturan lengkap pembuatan navigasi berbentuk tab ini ada di class AppSectionsPagerAdapter yang ada di MainActivity.java

3.3.3.3 Pembuatan Form Input Task

Pada halaman ini penulis akan membuat 2 inputan, inputan untuk task dan tanggal deadline, kemudian ada TextView tambahan untuk menampilkan informasi berapa hari lagi sampai deadline dari tanggal deadline yang penulis inputkan di field sebelumnya. Berikut listing program dari task_form_layout.xml

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10dp"
    tools:context=".TaskFormFragmentActivity">
    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:ems="10"
        android:gravity="top|left"
        android:inputType="textMultiLine"
        android:lines="5"
        android:scrollHorizontally="false"

        android:background="@android:drawable/editbox_background_norm
al">
    </EditText>
```

```

<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/editText1"
    android:layout_below="@+id/editText1"
    android:layout_marginTop="10dp"
    android:text="Set Deadline"
    android:onClick="showDatePickerDialog"/>

<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBottom="@+id/button1"
    android:layout_alignRight="@+id/editText1"
    android:layout_alignTop="@+id/button1"
    android:layout_toRightOf="@+id/button1"

    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text=""
    android:gravity="center_vertical"/>

</RelativeLayout>

```

Widget Button di kode di atas di gunakan untuk menampilkan inputan DatePicker. Kemudian selanjutnya penulis akan membuat class yang mengontrol tampilan layout task_form_layout.xml ini.

Setelah seperti biasa menghubungkan widget yang ada di xml ke object yang ada di class TaskFormFragmentActivity berikut kode untuk menyimpan data task yang penulis inputkan.

```

private void saveTask() {
    TaskSQLiteHelper db = new TaskSQLiteHelper(this);
    db.addTask(new Task(titleEt.getText().toString(), date,
false, false, ""));
    Toast.makeText(this, "Success", Toast.LENGTH_LONG).show();
    setResult(RESULT_OK, null);
    finish();
}

```

Sebelum membuat method di atas penulis harus membuat class TaskSQLiteHelper yang di dalamnya terdapat fungsi-fungsi untuk kebutuhan database mulai dari insert, update, delete dan query.

3.3.3.4 Pembuatan Form Pengaturan

Form pengaturan ini berfungsi untuk menyimpan pengaturan aplikasi, di dalam aplikasi ini penulis akan menambahkan beberapa pengaturan yaitu pengaturan waktu untuk menampilkan notifikasi, kemudian warna berdasarkan jumlah hari deadline terdekat. Berbeda dengan layout yang sebelumnya Android Framework telah menyediakan fitur sendiri untuk membuat preference, dan tampilan untuk preference ini tidak di letakkan di folder layout melainkan di folder xml. Berikut listing program dari preference.xml

```
<PreferenceScreen
xmlns:android="http://schemas.android.com/apk/res/android" >
    <PreferenceCategory android:title="REMINDERS" >
        <id.ac.gunadarma.tugasku.ui.TimePreference
            android:defaultValue="08:00"
            android:key="notif_time"
            android:summary="Set your desired time for the
reminder"
            android:title="Notification Time" />
    </PreferenceCategory>
    <PreferenceCategory android:title="URGENCY" >
        <id.ac.gunadarma.tugasku.ui.NumberPreference
            android:defaultValue="3"
            android:key="very_urgent"
            android:summary="Set urgency days"
            android:title="Very Urgent (Red)" />
        <id.ac.gunadarma.tugasku.ui.NumberPreference
            android:defaultValue="7"
            android:key="urgent"
            android:summary="Set urgency days"
            android:title="Urgent (Yellow)" />
    </PreferenceCategory>
</PreferenceScreen>
```

Tag yang di gunakan juga berbeda dengan widget yang sebelumnya penulis buat di layout. PreferenceScreen ini sebagai container utama, kemudian di dalamnya di bagi menjadi dua kelompok. TimePreference ini adalah class custom untuk

menampilkan input date di preference karena di Android belum menyediakan fitur DatePicker di preference, begitu juga dengan NumberPreference ini adalah class custom untuk menampilkan input jumlah hari. Untuk preference ini penulis tidak perlu membuat class Activity seperti biasa, penulis hanya membuat class Fragment seperti di bawah ini.

```
public class PrefsFragment extends PreferenceFragment {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.preferences);
    }
}
```

Kemudian untuk memanggil class fragment ini penulis menggunakan `getFragmentManager().beginTransaction().replace(android.R.id.content, new PrefsFragment()).commit();`

3.3.3.5 Membuat Database SQLite

Untuk membuat database di android penulis akan menggunakan implementasi dari class SQLiteOpenHelper di dalam class ini terdapat fungsi-fungsi Create, Read, Update dan Delete. Pertama penulis mendefinisikan struktur database yang akan penulis buat.

```
public static final int DATABASE_VERSION = 1;
public static final String DATABASE_NAME = "TaskDB";
public static final String TABLE_TASK = "task";
public static final String KEY_ID = "id";
public static final String KEY_TITLE = "title";
public static final String KEY_DEADLINE = "deadline";
public static final String KEY_DONE = "done";
public static final String KEY_DELETED = "deleted";
public static final String KEY_REMOTE_ID = "remote";
```

Di database ini penulis akan membuat table task dengan field id, title, deadline, done, deleted, remote. Table ini nanti yang akan penulis gunakan untuk

kebutuhan sinkronisasi dengan data yang ada di server yang telah penulis buat sebelumnya. Kemudian di method onCreate penulis akan mengisi syntax untuk membuat tabel Task.

```
String CREATE_TASK_TABLE =
    "CREATE TABLE "+TABLE_TASK+" ( "
    +KEY_ID+ " INTEGER PRIMARY KEY AUTOINCREMENT, "
    +KEY_TITLE+ " TEXT, "
    +KEY_DEADLINE+ " INTEGER, "
    +KEY_DONE+ " INTEGER, "
    +KEY_DELETED+ " INTEGER, "
    +KEY_REMOTE_ID+ " TEXT) ";
db.execSQL(CREATE_TASK_TABLE);
```

Kemudian selanjutnya penulis akan melengkapi method-method untuk kebutuhan ke database.

```
public void addTask(Task task){
    Log.d("addTask", task.toString());
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(KEY_TITLE, task.getTitle());
    Calendar to = Calendar.getInstance();
    to.setTime(new Date(task.getDeadline()));
    values.put(KEY_DEADLINE, to.getTimeInMillis());
    values.put(KEY_DONE, task.isDone() ? 1 : 0);
    values.put(KEY_DELETED, task.isDeleted() ? 1 : 0);
    values.put(KEY_REMOTE_ID, task.getRemoteId());
    db.insert(TABLE_TASK, null, values);
    db.close();
}
```

Di kode di atas adalah contoh untuk insert data ke dalam tabel yang sudah di buat sebelumnya.

3.3.3.6 Membuat Sync Adapter

SyncAdapter adalah sebuah fitur yang di sediakan oleh Android yang akan menghandle background syncs yang artinya mengambil atau mengirim data di proses dalam, biasanya di gunakan untuk mensinkronasikan data yang ada di android ke server. Fitur ini terdaftar di dalam SyncManager yang mengatur jalanya sinkronisasi. Berikut adalah keuntungan menggunakan SyncAdapter

1. Tampilan, semua SyncAdapter di android dapat di akses di menu Settings, di bawah Account. Di sini user di berikan beberapa pengaturan seperti mematikan sinkronisasi.
2. Mekanisme Retry, sebuah SyncAdapter terdapat implementasi untuk mengulang kembali sinkronisas jika sinkronisasi yang di lakukan sebelumnya gagal.

Berikut adalah step step untuk mengimplementasikan SyncAdapter di Android.

1. Membuat Account Authenticator dan ContentProvider di sini terdapat fungsi untuk membuat akun di android dan menghandle manipulasi data.
2. Membuat class SyncAdapter disini penulis akan membuat logika untuk sinkronisasi data ke server.
3. Membuat SyncService service yang menjalankan SyncAdapter penulis.

Sebuah SyncAdapter akan mengakses data yang terdapat di lokal, yang disediakan oleh ContentProvider dan mengakses server dengan mengirimkan token yang di dapat dari Authenticator. Berikut adalah listing program untuk mendapatkan token dari server, fungsi ini terdapat di class TaskServerAuthenticate

```
@Override
public String userSignIn(String username, String password, String
authType) throws Exception {
    DefaultHttpClient httpClient = new DefaultHttpClient();
    String url = "http://tugasku.herokuapp.com/login/token";
    HttpPost httpPost = new HttpPost(url);
```

```

        List<NameValuePair> nameValuePairs = new
ArrayList<NameValuePair>();
        nameValuePairs.add(new BasicNameValuePair("username",
username));
        nameValuePairs.add(new BasicNameValuePair("password",
password));

        httpPost.setEntity(new UrlEncodedFormEntity(nameValuePairs));

        String authToken = null;
        try {
            HttpResponse response = httpClient.execute(httpPost);
            String responseString =
EntityUtils.toString(response.getEntity());
            CredentialsResponse registerResponse = new
Gson().fromJson(responseString, CredentialsResponse.class);
            if (registerResponse.statusCode != 0) {
                throw new Exception("Error signing-in
["+registerResponse.statusCode+"] - " +
registerResponse.statusMessage);
            } else {
                authToken = registerResponse.secretToken;
            }
            Log.d("Task", "Response:
"+registerResponse.toString());
        } catch (IOException e) {
            e.printStackTrace();
        }
        return authToken;
    }
}

```

Di baris kode di atas pertama penulis akan login ke server dengan membawa parameter username dan password dengan menggunakan HttpPost kemudian memparsing response yang di terima, jika login sukses maka response yang di terima adalah token yang kemudian di gunakan untuk mengakses data di server tanpa harus login terlebih dahulu.

Di dalam content provider ini terdapat fungsi yang menjembatani antara SyncAdapter dengan database, karena penulis telah membuat implementasi SqliteOpenHelper maka akan lebih mudah membuat Contentprovider ini, berikut contoh method untuk insert data di Contentprovider ke database melalui SqliteOpenHelper


```

@Override
public Uri insert(Uri uri, ContentValues values) {
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    int token = URI_MATCHER.match(uri);
    switch (token) {
        case PATH_TOKEN: {
            long id = db.insert(TaskSQLiteHelper.TABLE_TASK, null,
values);
            if (id != -1)

                getContext().getContentResolver().notifyChange(uri, null);
            return
CONTENT_URI.buildUpon().appendPath(String.valueOf(id)).build();
        }
        default: {
            throw new UnsupportedOperationException("URI: " +
uri + " not supported.");
        }
    }
}

```

Setelah selesai maka penulis akan membuat class SyncAdapter yang mengatur logika untuk sinkronisasi

Fungsi utama di dalam class ini terdapat di method onPerformSync method ini yang akan di jalankan ketika tiba jadwal untuk sinkronisasi yang telah di jadwalkan oleh system.

```

@Override
public void onPerformSync(Account account, Bundle extras, String
authority,
    ContentProviderClient provider, SyncResult syncResult) {
    StringBuilder sb = new StringBuilder();
    if (extras != null) {
        for (String key : extras.keySet()) {
            sb.append(key + "[" + extras.get(key) + "] ");
        }
    }
    Log.d(TAG, "onPerformSync for account[" + account.name + "],
Extras: " + sb.toString());
    try {
        String authToken =
mAccountManager.blockingGetAuthToken(account,
AccountManager.KEY_AUTHTOKEN, true);
        String userId =
mAccountManager.getUserData(account,

```

```

AccountGeneral.USERDATA_USER_OBJ_ID);
        TaskSQLiteHelper sqliteHelper = new
TaskSQLiteHelper(getContext());
        Log.d(TAG, "onPerformSync [" + authToken + "].
userObjectId: " + userObjectId);
        List<id.ac.gunadarma.tugasku.helper.dao.Task>
remoteTaskList = Api.getTaskList(authToken);
        Log.d(TAG, "onPerformSync, Remote Task: " +
remoteTaskList.toString());
        ArrayList<Task> localTaskList = new ArrayList<Task>();
        Cursor curTvShows =
provider.query(TaskContentProvider.CONTENT_URI, null, null, null,
null);

        if (curTvShows != null) {
            while (curTvShows.moveToNext()) {

                localTaskList.add(Task.fromCursor(curTvShows));
            }
            curTvShows.close();
        }
        Log.d(TAG, "onPerformSync, Local Task: " +
localTaskList.toString());
        ArrayList<Task> taskToRemote = new ArrayList<Task>();
        for (Task localTask : localTaskList) {
            if("").equals(localTask.getRemoteId())
                taskToRemote.add(localTask);
        }
        Log.d(TAG, "onPerformSync, Task to Remote: " +
taskToRemote.toString());
        ArrayList<Task> taskToLocal = new ArrayList<Task>();
        for (id.ac.gunadarma.tugasku.helper.dao.Task remoteTask
: remoteTaskList) {
            if(!sqliteHelper.isSynced(remoteTask.id))
                taskToLocal.add(remoteTask.getTaskLocal());
        }
        Log.d(TAG, "onPerformSync, Task to Local: " +
taskToLocal.toString());
        if (taskToRemote.size() == 0) {
            Log.d("Task", TAG + "> No local changes to update
server");
        } else {
            Log.d("Task", TAG + "> Updating remote server
with local changes");
            for (Task remoteTask : taskToRemote) {
                Log.d("Task", TAG + "> Local -> Remote [" +
remoteTask.getId() + "]");
                String id =
Api.postTask(remoteTask.getTaskRemote(), authToken);
                if(!"".equals(id)){

                    sqliteHelper.markSynced(remoteTask.getId(), id);
                }
            }
        }
    }
}

```

```

        }
        if (taskToLocal.size() == 0) {
            Log.d("Task", TAG + "> No server changes to
update local database");
        } else {
            Log.d("Task", TAG + "> Updating local database
with remote changes");
            int i = 0;
            ContentValues showsToLocalValues[] = new
ContentValues[taskToLocal.size()];
            for (Task localTask : taskToLocal) {
                Log.d("Task", TAG + "> Remote -> Local [" +
localTask.getRemoteId() + "]");
                showsToLocalValues[i++] =
localTask.getContentValues();
            }

            provider.bulkInsert(TaskContentProvider.CONTENT_URI,
showsToLocalValues);
            Log.d("Task", TAG + "> Finished.");

        } catch (OperationCanceledException e) {
            e.printStackTrace();
        } catch (IOException e) {
            syncResult.stats.numIoExceptions++;
            e.printStackTrace();
        } catch (AuthenticatorException e) {
            syncResult.stats.numAuthExceptions++;
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

Di dalam method di atas pertama penulis mengambil token yang sudah penulis dapatkan pada proses login sebelumnya kemudian mengambil data dari server dan dari lokal. Kemudian mensinkronasikan keduanya data mana yang harus di upload ke server atau sebaliknya data mana yang harus di insert ke lokal database agar data bisa sama antara server dan lokal.

Setelah selesai membuat SyncAdapter selanjutnya penulis akan mendaftarkan SyncService agar SyncAdapter yang telah penulis buat di kenali oleh system.

```

<provider android:authorities="id.ac.gunadarma.tugasku.provider"
    android:name=".db.TaskContentProvider"
    android:label="@string/provider_name"
    android:exported="false"/>

<service android:name=".sync.TaskSyncService"
    android:exported="true" >
    <intent-filter>
        <action android:name="android.content.SyncAdapter" />
    </intent-filter>
    <meta-data android:name="android.content.SyncAdapter"
    android:resource="@xml/sync_adapter" />
</service>

<service android:name=".account.TaskAuthenticatorService">
    <intent-filter>
        <action
    android:name="android.accounts.AccountAuthenticator" />
    </intent-filter>
    <meta-data
    android:name="android.accounts.AccountAuthenticator"
    android:resource="@xml/authenticator" />
</service>

```

Kode xml di atas adalah konfigurasi agar Service yang telah penulis buat di kenali oleh android yang kemudian akan di jalankan otomatis.

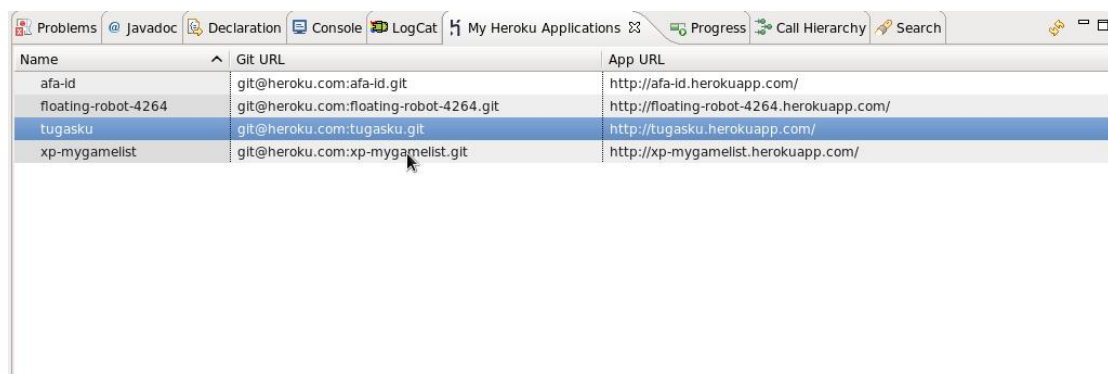
3.4 Deploy Aplikasi

Heroku adalah layanan cloud, tepatnya Platform as a service (PaaS). Heroku membuat penulis dapat fokus hanya untuk pembuatan aplikasi tanpa mengkhawatirkan masalah server seperti instalasi dan maintenance yang semuanya sudah dihandle oleh Heroku. Selain itu, penulis dapat membuat akun gratis di Heroku. Sebuah free account di Heroku dapat hosting maksimal 5 aplikasi.

Di samping itu, Heroku mempunyai konsep Add-ons, dimana penulis dapat menambah berbagai macam teknologi tambahan seperti PostgreSQL, MongoDB, MySQL, Redis, Hadoop, dan masih banyak lagi, ke dalam aplikasi penulis yang

Sedang berjalan secara on the fly. Bahkan beberapa add-ons tersebut ada yang free. Heroku juga mempunyai Heroku Toolbelt, yaitu semacam aplikasi console (CLI) untuk konfigurasi aplikasi yang penulis di Heroku. Penulis dapat menggunakan Heroku Toolbelt untuk login, membuat aplikasi, mendeploy, memasang addons, dll.

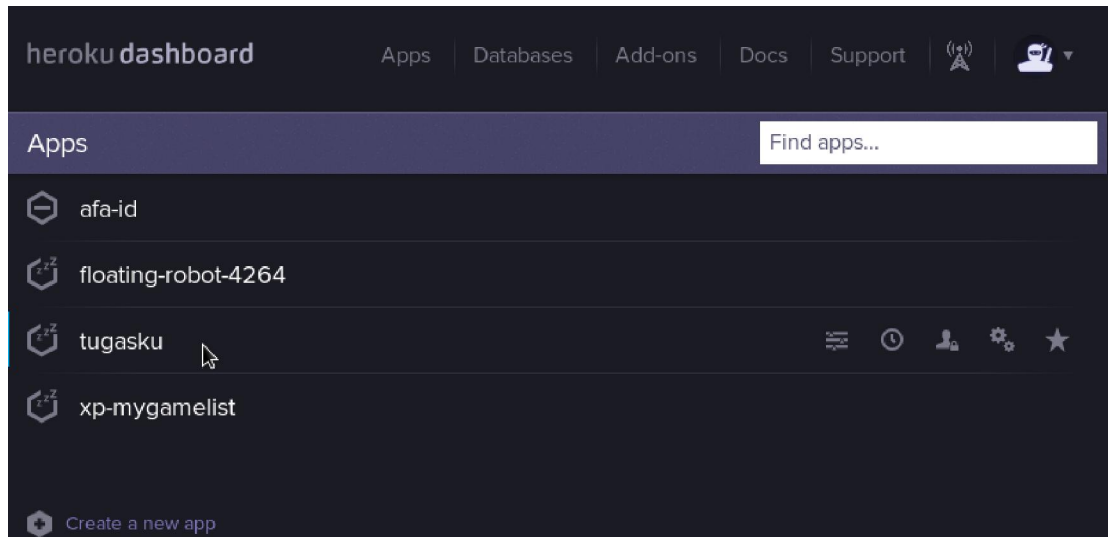
Ketika penulis membuat project aplikasi web menggunakan Heroku Plugin di eclipse, maka akan otomatis aplikasi penulis akan di buatkan git repository di lokal dan di server heroku. Project ini bisa penulis lihat di folder `~/git/heroku/{nama-project-penulis}`. Kemudian untuk melihat aplikasi apa saja yang sudah penulis buat penulis dapat melihat di eclipse bagian My Heroku Applications



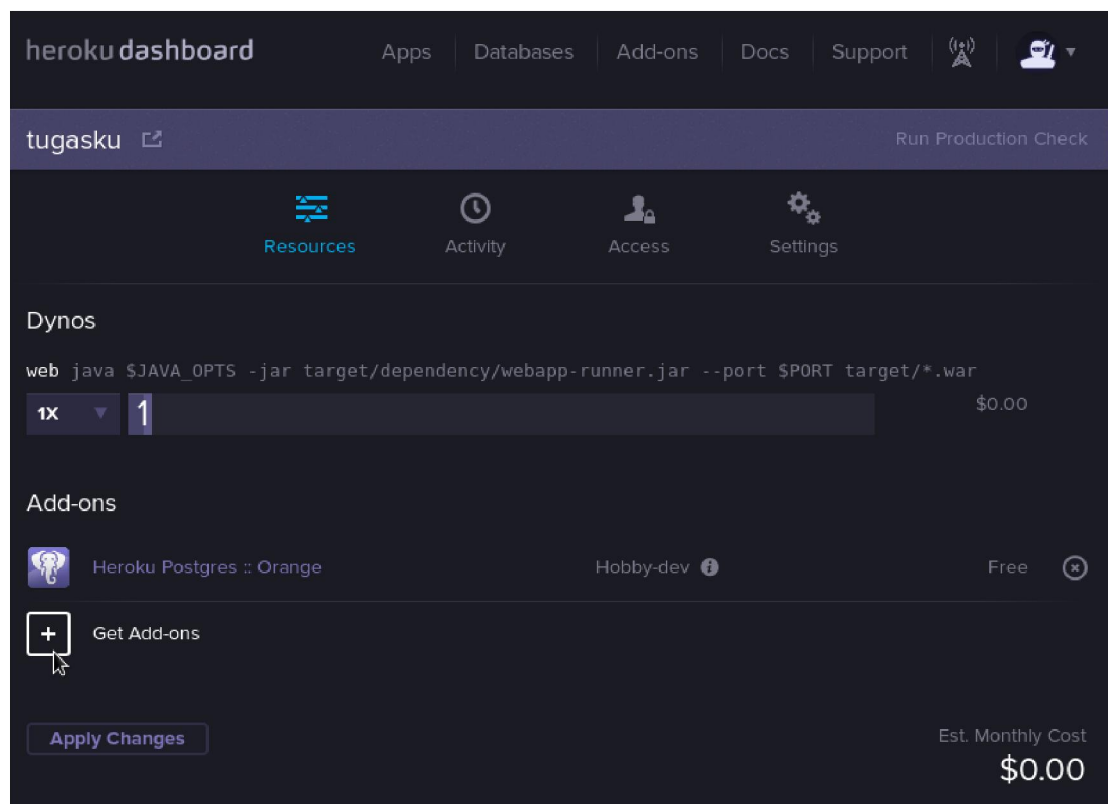
Name	Git URL	App URL
afa-id	git@heroku.com:afa-id.git	http://afa-id.herokuapp.com/
floating-robot-4264	git@heroku.com:floating-robot-4264.git	http://floating-robot-4264.herokuapp.com/
tugasku	git@heroku.com:tugasku.git	http://tugasku.herokuapp.com/
xp-mygamelist	git@heroku.com:xp-mygamelist.git	http://xp-mygamelist.herokuapp.com/

Pertama sebelum penulis mendeploy aplikasi ke Heroku Cloud penulis harus melakukan beberapa konfigurasi terlebih dahulu, konfigurasi pertama adalah penulis akan menambahkan plugin database PostgreSQL ke dalam aplikasi penulis. Berikut langkah-langkahnya

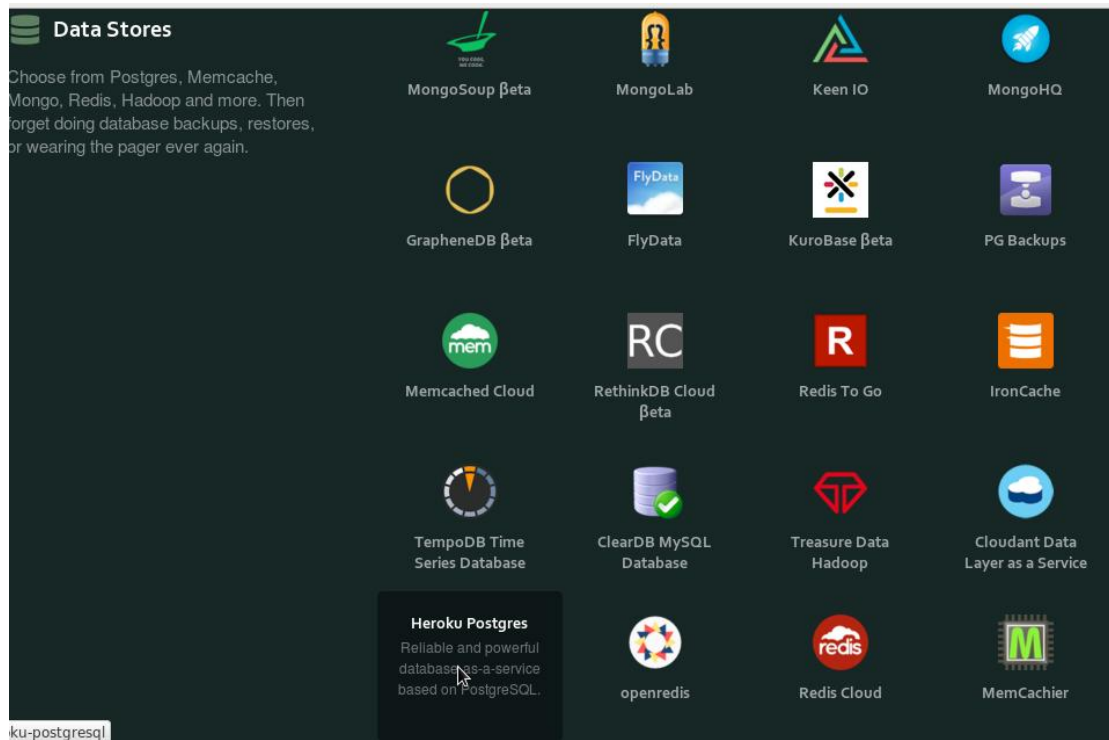
1. Login ke halaman dashboard heroku.
2. Pilih aplikasi yang akan penulis tambahkan database PostgreSQL



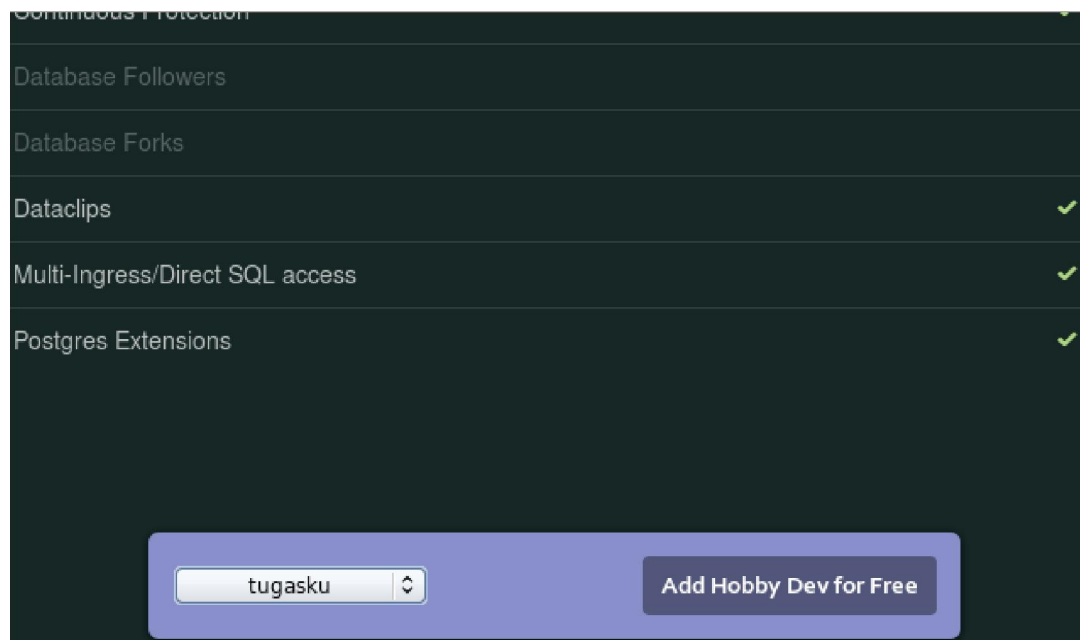
3. Di Bagian konfigurasi aplikasi pilih menu Get Add-Ons



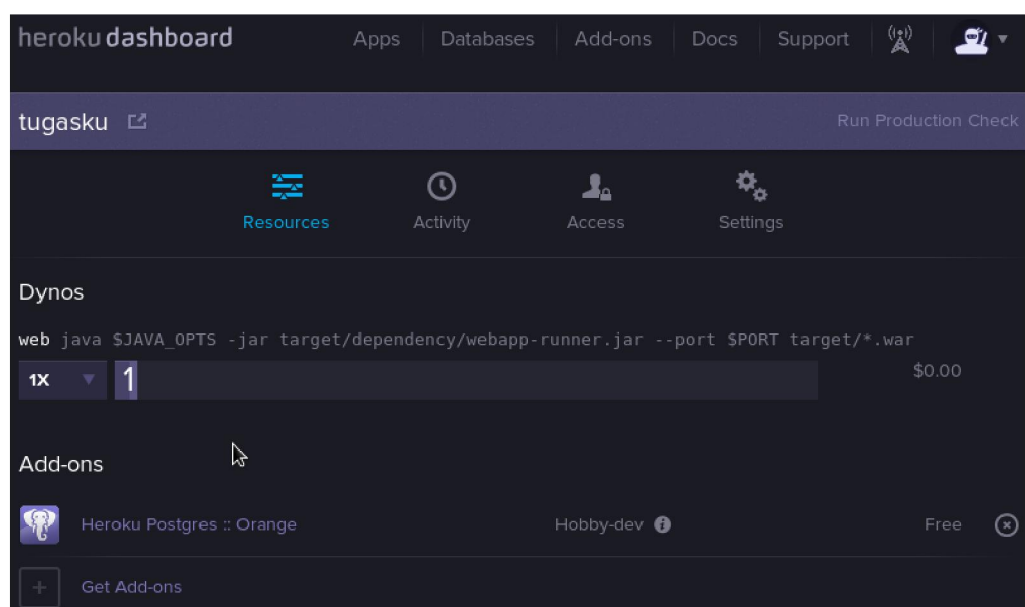
4. di halaman ini penulis dapat memilih Add-ons sesuai yang penulis butuhkan. Karena aplikasi penulis akan menggunakan database postgresql maka di sini penulis akan memilih add-ons Heroku Postgres.



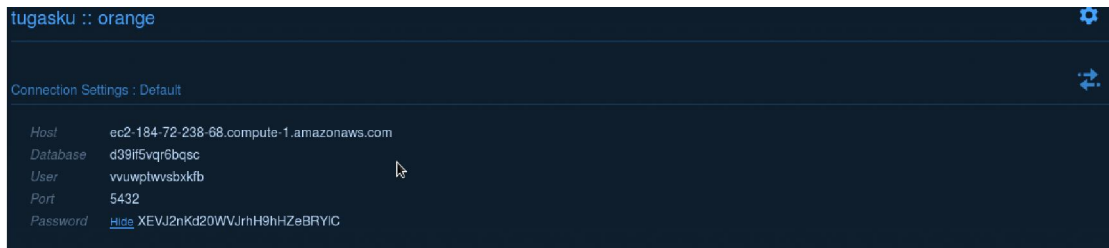
5. Setelah itu penulis akan masuk ke halaman pricing plans, di sini penulis bisa memilih paket sesuai kebutuhan. Untuk penulisan ini penulis akan menggunakan plan Free atau gratis, kemudian di bawah ada pilihan aplikasi mana yang akan penulis tambahkan Add-on ini. Setelah selesai klik Add Hobby Dev for Free.



6. Setelah sukses menambahkan add-on maka secara otomatis Heroku akan membuat database di server Heroku, untuk melihat lebih detail informasi database penulis bisa mengakses ke dashboard di bagian add-on klik Heroku Postgres



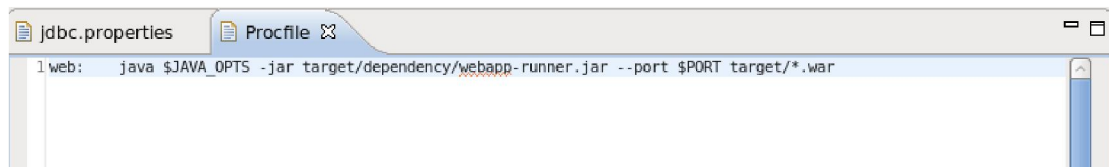
7. di halaman ini adalah letak di mana konfigurasi untuk database penulis. selanjutnya penulis akan memasang konfigurasi ini ke aplikasi penulis



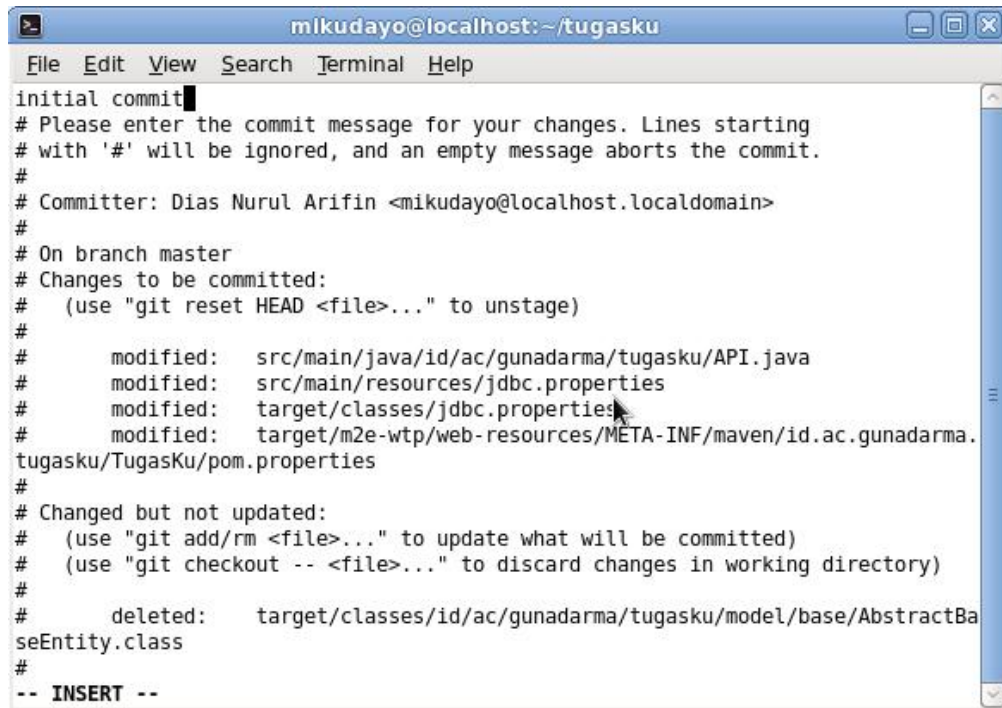
8. di file konfigurasi jdbc.properties di project penulis masukkan konfigurasi database seperti di step sebelumnya



9. setelah konfigurasi database selesai selanjutnya penulis akan mengenalkan kepada Heroku aplikasi yang telah penulis buat adalah aplikasi web dan file mana yang akan di deploy di file Procfile



10. Selanjutnya penulis akan mendeploy dan menjalankan aplikasi. Masuk ke folder aplikasi penulis dan commit dengan perintah git commit.



```
mikudayo@localhost:~/tugasku
File Edit View Search Terminal Help
initial commit
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Committer: Dias Nurul Arifin <mikudayo@localhost.localdomain>
#
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   src/main/java/id/ac/gunadarma/tugasku/API.java
#       modified:   src/main/resources/jdbc.properties
#       modified:   target/classes/jdbc.properties
#       modified:   target/m2e-wtp/web-resources/META-INF/maven/id.ac.gunadarma.
tugasku/TugasKu/pom.properties
#
# Changed but not updated:
#   (use "git add/rm <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       deleted:    target/classes/id/ac/gunadarma/tugasku/model/base/AbstractBa
seEntity.class
#
-- INSERT --
```

11. Setelah semua source code di commit ke repository lokal maka selanjutnya penulis akan melakukan push ke repository yang ada di heroku server dengan perintah git push. Setelah sukses push ke repository server maka heroku akan otomatis men-compile dan men-deploy aplikasi penulis.

```
mikudayo@localhost:~/tugasku
File Edit View Search Terminal Help
[mikudayo@localhost tugasku]$ git push
Fetching repository, done.
Counting objects: 41, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (13/13), done.
Writing objects: 100% (21/21), 1.49 KiB, done.
Total 21 (delta 5), reused 0 (delta 0)

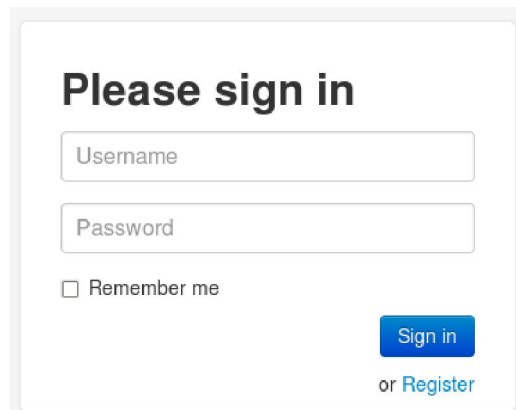
-----> Java app detected
-----> Installing OpenJDK 1.6... done
-----> Installing settings.xml... done
-----> executing /app/tmp/cache/.maven/bin/mvn -B -Duser.home=/tmp/build_2771ae1e-b114-47bb-9f42-8254aa6b96f3 -Dmaven.repo.local=/app/tmp/cache/.m2/repository -s /app/tmp/cache/.m2/settings.xml -DskipTests=true clean install
[INFO] Scanning for projects...
[WARNING]
[WARNING] Some problems were encountered while building the effective model
el for id.ac.gunadarma.tugasku:Tugasku:war:1.0-SNAPSHOT
[WARNING] 'dependencies.dependency.(groupId:artifactId:type:classifier)'
must be unique: postgresql:postgresql:jar -> version 8.3-606.jdbc4 vs 9.1-901.jd
bc4 @ line 122, column 15
[WARNING] 'dependencies.dependency.(groupId:artifactId:type:classifier)'
must be unique: commons-fileupload:commons-fileupload:jar -> duplicate declarati
on of version 1.2.2 @ line 133, column 15
```

```
mikudayo@localhost:~/tugasku
File Edit View Search Terminal Help
- SNAPSHOT/Tugasku-1.0-SNAPSHOT.war
[INFO] Installing /tmp/build_2771ae1e-b114-47bb-9f42-8254aa6b96f3/pom.xml
to /app/tmp/cache/.m2/repository/id/ac/gunadarma/tugasku/Tugasku/1.0-SNAPSHOT/T
ugasku-1.0-SNAPSHOT.pom
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 6.997s
[INFO] Finished at: Mon Mar 31 11:58:44 UTC 2014
[INFO] Final Memory: 16M/513M
[INFO] -----
-----> Discovering process types
Procfile declares types -> web
-----> Compressing... done, 88.0MB
-----> Launching... done, v63
http://tugasku.herokuapp.com/ deployed to Heroku

To git@heroku.com:tugasku.git
4c6607c..01ca83a master -> master
[mikudayo@localhost tugasku]$
```

3.5 Demo Aplikasi

Untuk menggunakan aplikasi Management tugas ini langkah pertama adalah register akun di server yang sudah penulis deploy di Heroku. Di dalam penulisan ini URL aplikasi yang sudah di deploy adalah <http://tugasku.herokuapp.com> setelah mengakses url tersebut maka akan tampil halaman login.



Please sign in

Username

Password

☐ Remember me

[Sign in](#)

or [Register](#)

Untuk memperoleh akun klik link register.

Manajemen Tugas



First Name Tugasku

Last Name

Username tugasku

Password

Email tugasku@gmail.com

[Register](#)

Setelah selesai mengisi data akun, klik tombol register

Registered!! Please Sign In Below

Please sign in

Username

Password

☐ Remember me

Sign in

or [Register](#)

Setelah berhasil register akun, selanjutnya penulis bisa langsung login dengan akun yang telah penulis buat

Management Tugas

User

Tasks

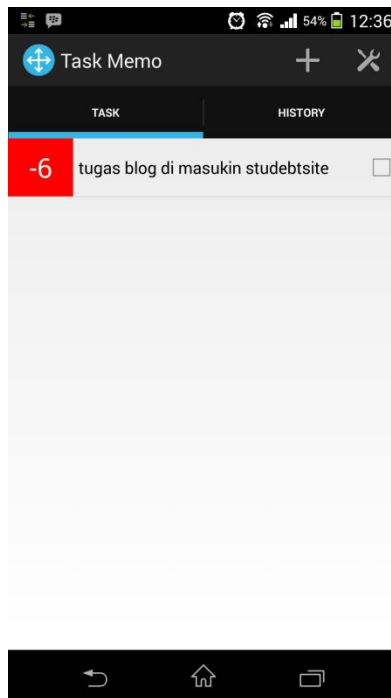
#	Content	Deadline	Task Done?
---	---------	----------	------------

Found 0 row(s)

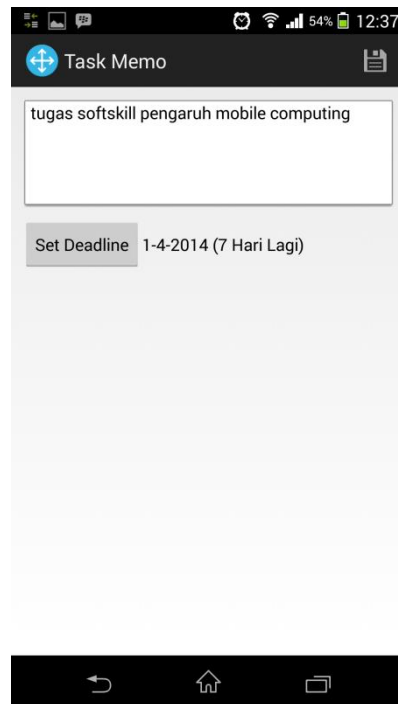
1 of 0

Ketika penulis login ke dalam aplikasi penulis akan di di arahkan ke halaman index dimana halaman ini yang nantinya akan menampilkan tugas yang telah penulis input di aplikasi android. Untuk selanjutnya penulis akan menjalankan aplikasi di android.

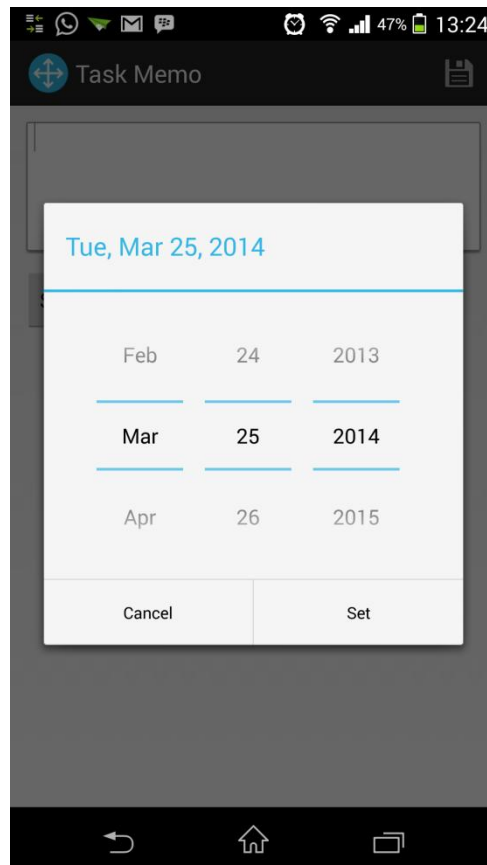
Ketika penulis membuka aplikasi android Management Tugas ini akan langsung menampilkan data tugas penulis, dan data history dari tugas yang sudah selesai.



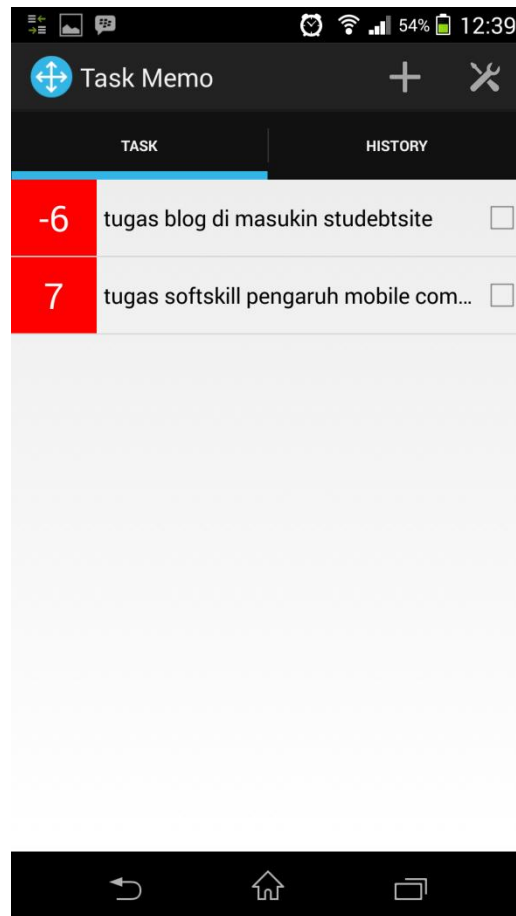
Untuk menambahkan tugas penulis bisa mengakses dengan menu tambah di pojok kiri atas.



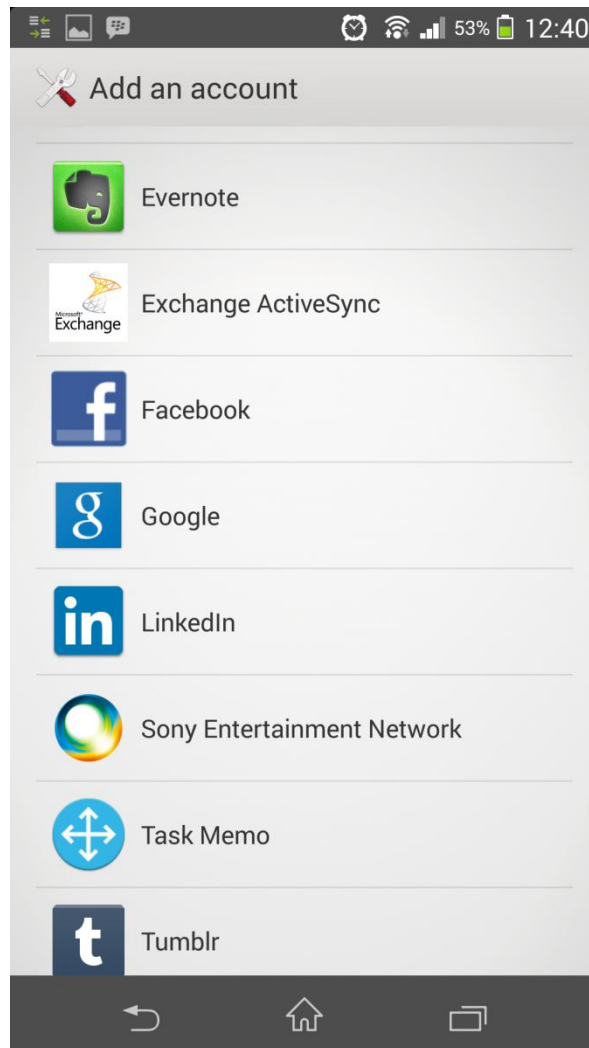
Setelah tampil form isi data, penulis bisa mengatur deadline dengan tombol Set Deadline



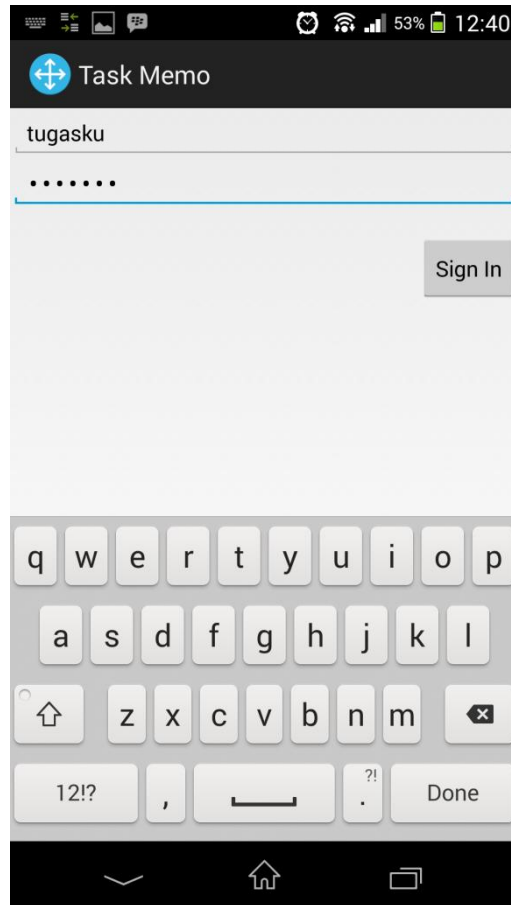
Kemudian pilih pilih menu save di pojok kiri atas. Setelah selesai maka penulis akan di arahkan ke halaman home dengan tambahan data baru yang barusaja penulis input.



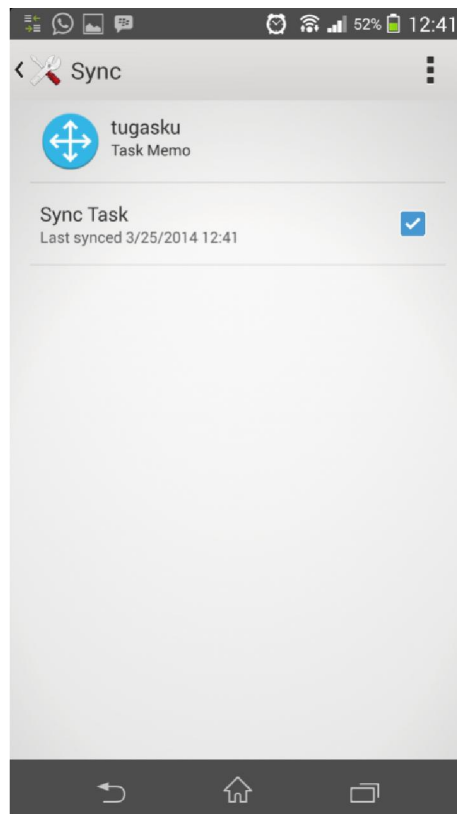
Untuk mengaktifkan sinkronisasi data ke server cloud yang sebelumnya sudah di deploy. Di android penulis perlu login dengan akun yang sudah teregister. Di handphone Android masuk ke menu setting > account



Kemudian pilih account Task Memo



Setelah itu penulis akan di minta login dengan akun yang sudah terdaftar. Ketika login telah berhasil maka data yang telah penulis isi di lokal akan otomatis tersinkronisasi dengan data di server.



Setelah status sudah selesai tersinkronisasi penulis bisa cek di server dengan masuk ke aplikasi management tugas versi web.

Management Tugas

User

Tasks

#	Content	Deadline	Task Done?
1	tugas softskill pengaruh mobile computing	Apr 1, 2014 12:00:00 AM	no

Found 1 row(s)

1 of 1

Di aplikasi Management tugas ini terdapat notifikasi dimana setiap waktu yang di kondigurasi di menu setting akan menampilkan notifikasi setiap hari untuk mengingatkan user.

