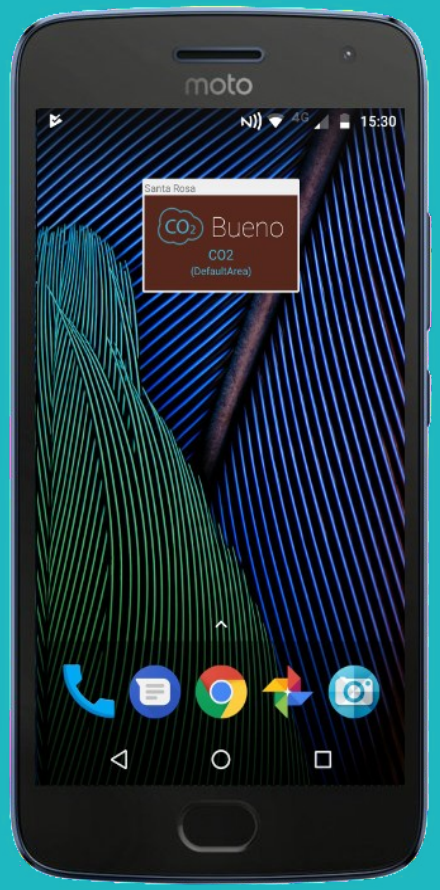
enControl widget

Realizado por Jordi Tejedo Jiménez

Dirigido por Francesc d'Assís Giralt Queralt

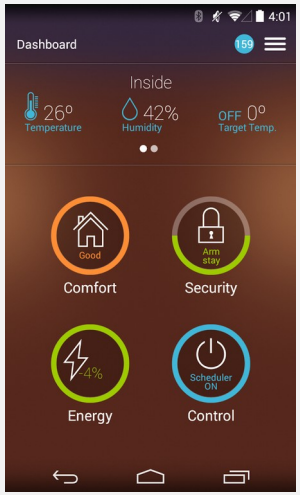


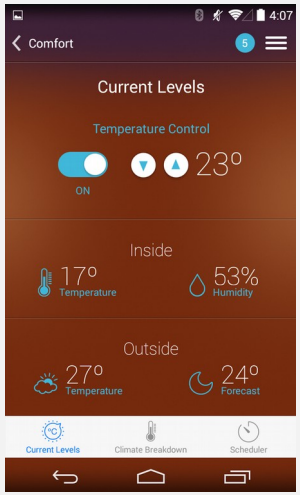
UOC
Universitat Oberta de Catalunya

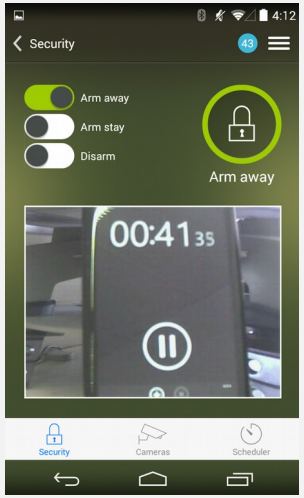
ÍNDICE

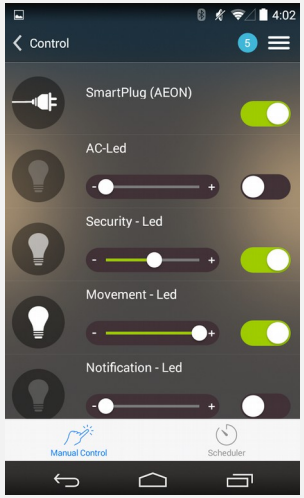
INTRODUCCIÓN

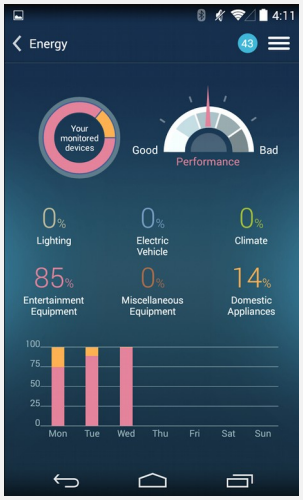
enControl es una solución de *smarthome* desarrollada íntegramente en Barcelona por la empresa Sensing & Control S.L.





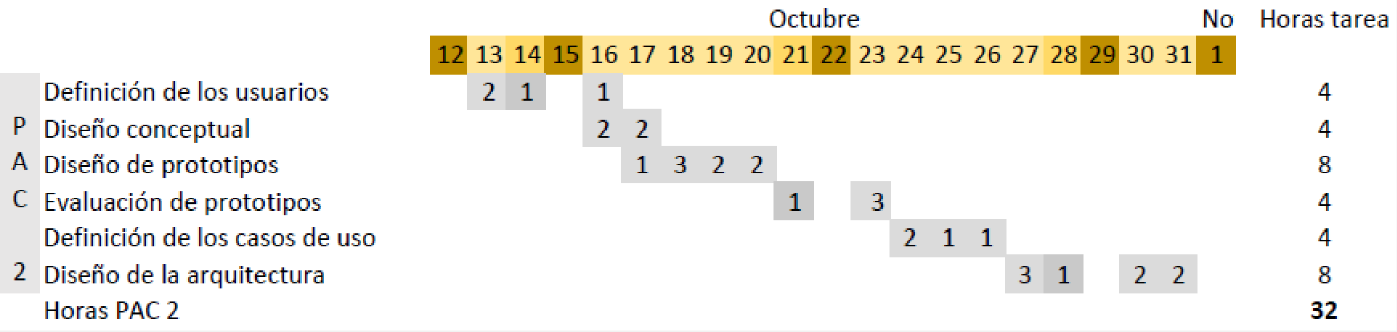




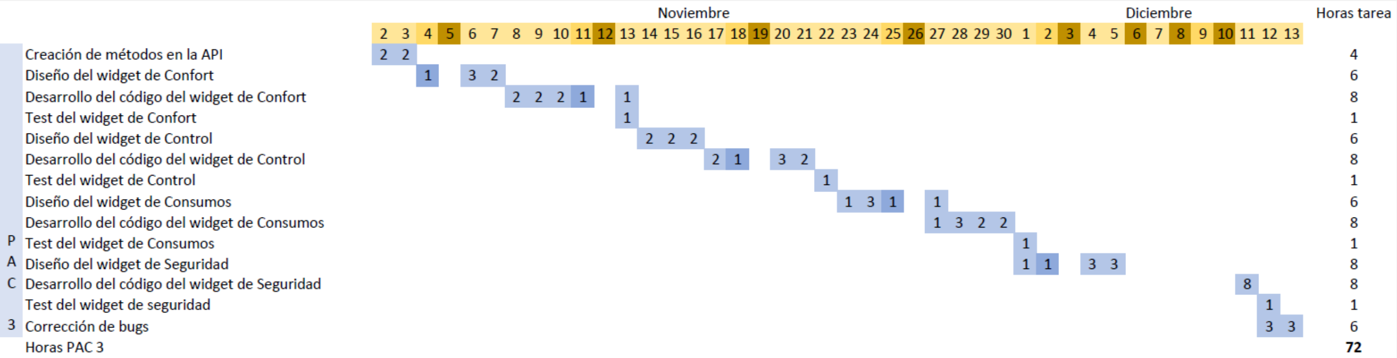


Contexto y justificación

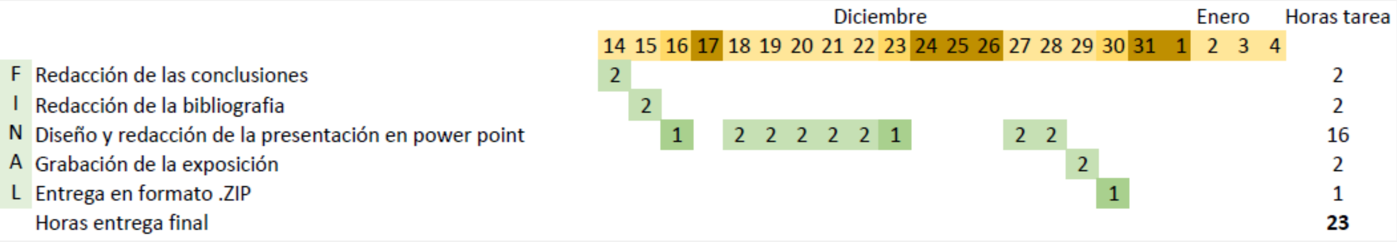
Planificación



		Octubre																				No	Horas tarea
		12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1	
	Definición de los usuarios		2	1		1																	4
P	Diseño conceptual					2	2																4
A	Diseño de prototipos						1	3	2	2													8
C	Evaluación de prototipos										1		3										4
	Definición de los casos de uso													2	1	1							4
2	Diseño de la arquitectura																3	1		2	2		8
	Horas PAC 2																						**32**

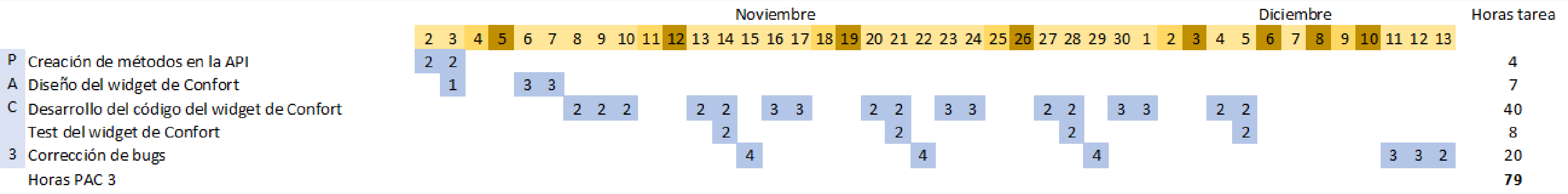


		Noviembre																												Diciembre													Horas tarea
		2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	1	2	3	4	5	6	7	8	9	10	11	12	13	
	Creación de métodos en la API	2	2																																									4
	Diseño del widget de Confort			1		3	2																																					6
	Desarrollo del código del widget de Confort							2	2	2	1		1																															8
	Test del widget de Confort												1																															1
	Diseño del widget de Control													2	2	2																												6
	Desarrollo del código del widget de Control																2	1		3	2																							8
	Test del widget de Control																					1																						1
	Diseño del widget de Consumos																						1	3	1		1																	6
	Desarrollo del código del widget de Consumos																										1	3	2	2														8
P	Test del widget de Consumos																														1													1
A	Diseño del widget de Seguridad																														1	1		3	3									8
C	Desarrollo del código del widget de Seguridad																																								8			8
	Test del widget de seguridad																																									1		1
3	Corrección de bugs																																									3	3	6
	Horas PAC 3																																											**72**



		Diciembre																		Enero				Horas tarea
		14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1	2	3	4	
F	Redacción de las conclusiones	2																						2
I	Redacción de la bibliografia		2																					2
N	Diseño y redacción de la presentación en power point			1		2	2	2	2	2	1				2	2								16
A	Grabación de la exposición																2							2
L	Entrega en formato .ZIP																	1						1
	Horas entrega final																							**23**

Planificación



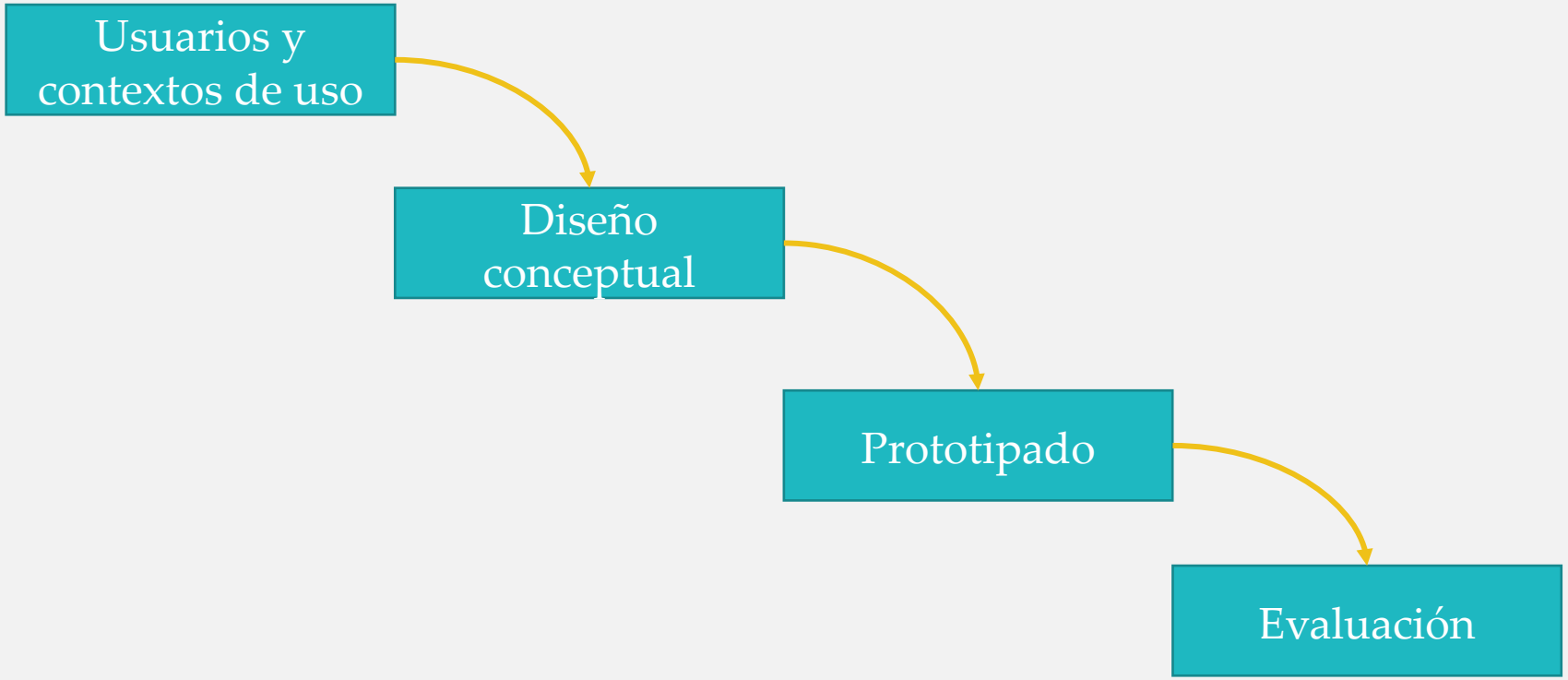
		Noviembre																													Diciembre													Horas tarea
		2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	1	2	3	4	5	6	7	8	9	10	11	12	13	
P	Creación de métodos en la API	2	2																																									4
A	Diseño del widget de Confort		1			3	3																																					7
C	Desarrollo del código del widget de Confort							2	2	2			2	2		3	3			2	2		3	3			2	2		3	3			2	2									40
	Test del widget de Confort													2							2							2							2									8
3	Corrección de bugs														4							4							4												3	3	2	20
	Horas PAC 3																																											**79**

Desarrollo → De 8 a 40 horas

Test → De 1 a 8 horas

Corrección de bugs → De 6 a 20 horas

DISEÑO CENTRADO EN EL USUARIO



Usuarios

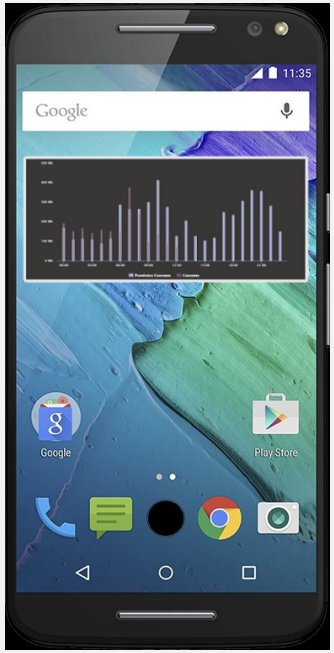
	Propietarios de inmuebles	Usufructuarios
Focales	X	X
Secundarios		X
No prioritarios		X
Promotores	X	

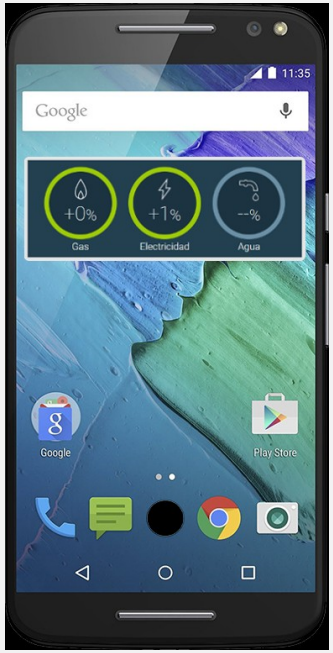
	Usuario
Conocimientos tecnológicos	Medios
Rango de edad	14 – 70 años
Gustos	Muy diversos
Ejemplos	Padres o madres de familia, hijos, dueños de negocios, clientes de un hotel…

Prototipos

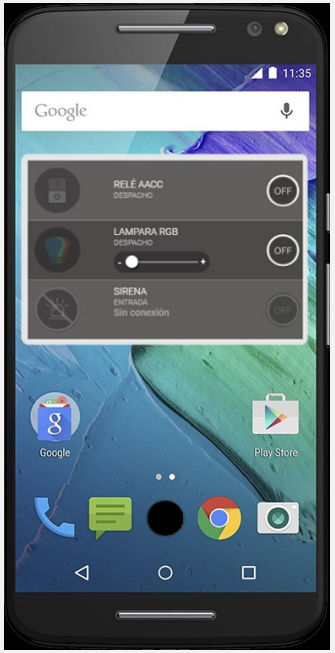
Confort

Seguridad



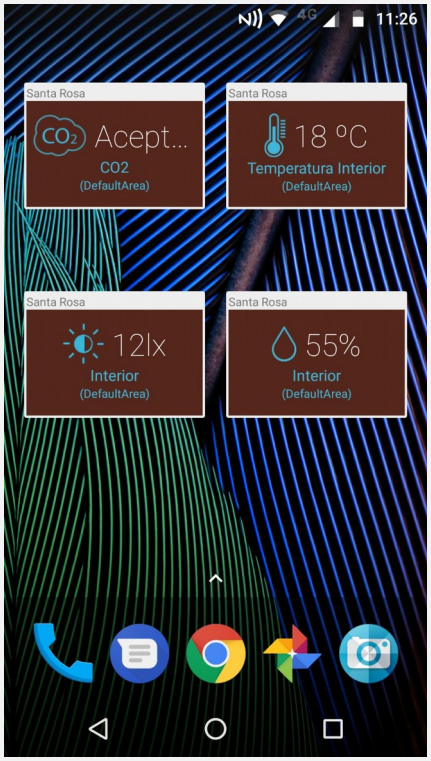


Consumos



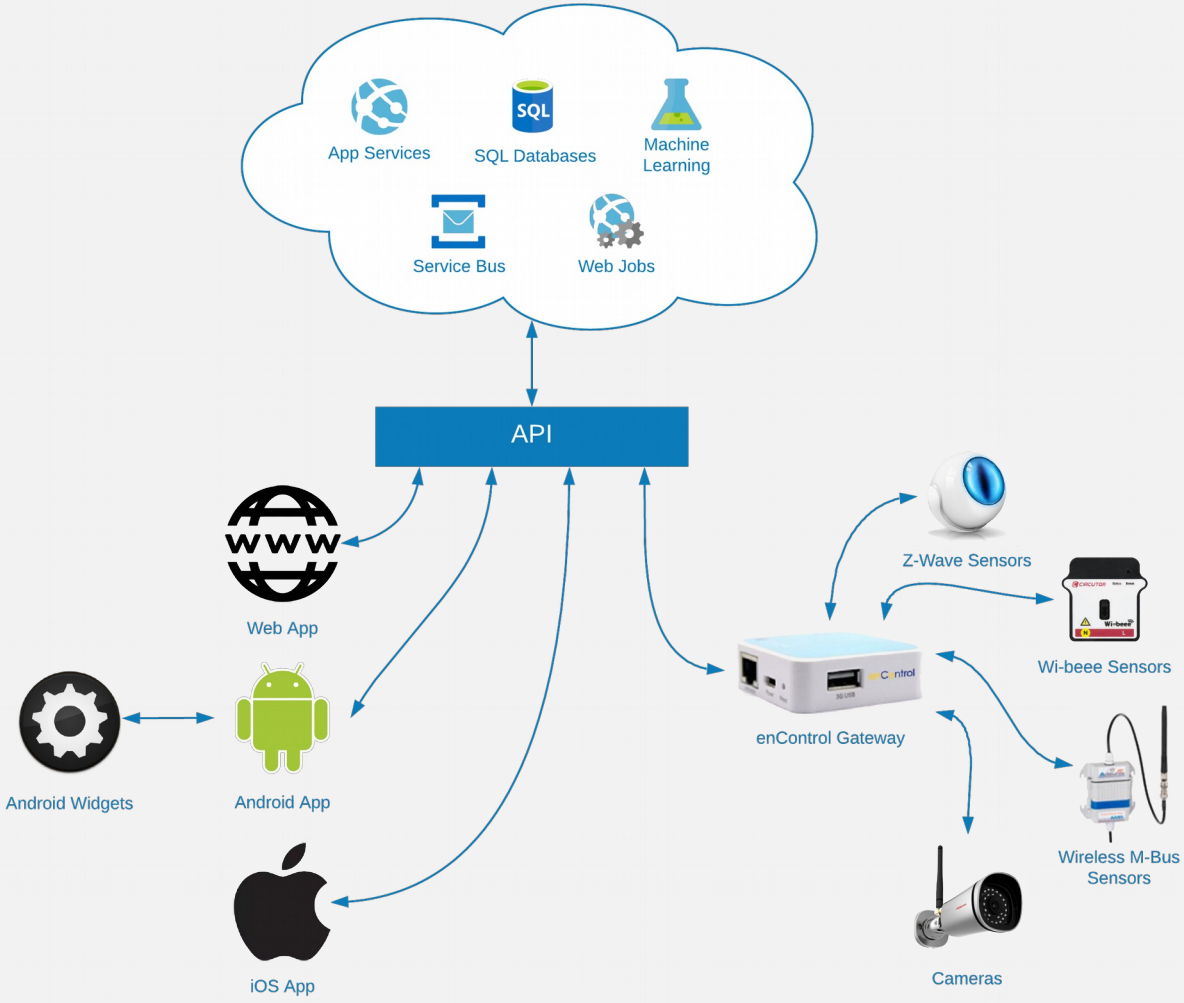
Control

Prototipos

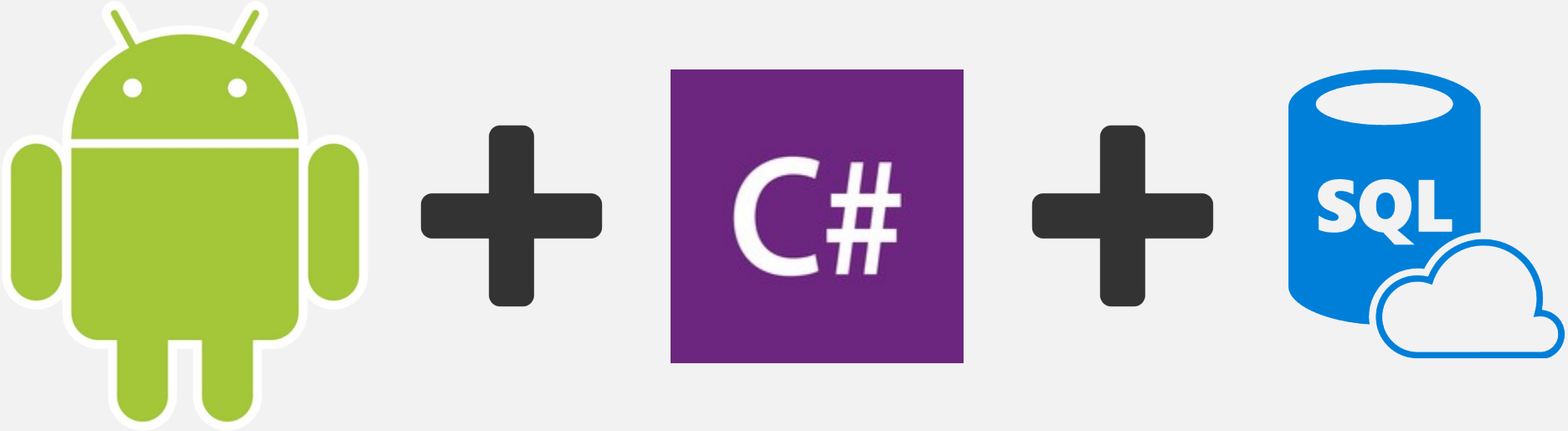


- Un solo sensor por *widget*
- Se permiten valores más largos
- Se muestra el area del sensor
- Se muestra la instalación del sensor

ARQUITECTURA DEL SISTEMA



DISEÑO SOFTWARE



Desarrollo

1. App Widget Provider Info
   - Archivo de propiedades del *widget*
   - Define:
     - Tamaño
     - Frecuencia de actualización
     - Localización del *layout*

- Actividad de configuración
- Modo de ajuste
- Categoría
- Imágen predeterminada

```
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:minWidth="110dp"
    android:minHeight="110dp"
    android:updatePeriodMillis="0"
    android:initialLayout="@layout/widget_comfort"
    android:configure="com.sensingcontrol.android.widgets.ComfortWidgetConfiguratio
nActivity"
    android:resizeMode="none"
    android:widgetCategory="home_screen"
    android:previewImage="@drawable/comfort_widget_preview">
</appwidget-provider>
```

Desarrollo

2. App Widget Provider

- Define los métodos básicos que permitan interactuar mediante programación con él *widget* en función de los eventos capturados

```
@Override
public void onUpdate(Context context, AppWidgetManager appWidgetManager,
int[] appWidgetIds) {
    final int N = appWidgetIds.length;

    SharedPreferences prefs =
context.getSharedPreferences("ComfortWidgetPrefs", Context.MODE_PRIVATE);

    // Perform this loop procedure for each App Widget that belongs to this provider
   for (int i=0; i<N; i++) {
        int appWidgetId = appWidgetIds[i];

        // Before update, check if appWidgetId exist
      if(prefs.contains(appWidgetId + "_id")) {
            updateWidget(context, appWidgetManager, appWidgetId);
        }
    }
}
```

Desarrollo

```java
@Override
public void onEnabled(Context context){
    LocalBroadcastManager.getInstance(context).registerReceiver(mSignalRReceiver, new IntentFilter("SignalRBroadcastMessage"));
}
```

```java
@Override
public void onDisabled(Context context){
    LocalBroadcastManager.getInstance(context).unregisterReceiver(mSignalRReceiver);

    SharedPreferences prefs = context.getSharedPreferences("ComfortWidgetPrefs", Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = prefs.edit();
    editor.clear().commit();
}
```

```java
@Override
public void onDeleted(Context context, int[] appWidgetIds) {
    // Search all appWidgetId info saved in SharedPreferences and remove
   for (int i=0; i < appWidgetIds.length; i++) {
        SharedPreferences prefs = context.getSharedPreferences("ComfortWidgetPrefs", Context.MODE_PRIVATE);
        String sensorId = prefs.getString(appWidgetIds[i] + "_id", "");

        SharedPreferences.Editor editor = prefs.edit();
        editor.remove(sensorId + "_mAppWidgetId");
        editor.remove(appWidgetIds[i] + "_installationName");
        editor.remove(appWidgetIds[i] + "_id");
        editor.remove(appWidgetIds[i] + "_name");
        editor.remove(appWidgetIds[i] + "_area");
        editor.remove(appWidgetIds[i] + "_type");
        editor.remove(appWidgetIds[i] + "_lastValue");
        editor.remove(appWidgetIds[i] + "_selected");
        editor.remove(appWidgetIds[i] + "_baseIconName");
        editor.remove(appWidgetIds[i] + "_units");
        editor.commit();
    }
}
```

Desarrollo

3. Interfaz gráfica
   - Icono del tipo de sensor (ImageView)
   - Nombre (TextView)
   - Valor de la medida (TextView)
   - Area (TextView)
   - Instalación (TextView)

Desarrollo

4. Android Manifest

- Receiver

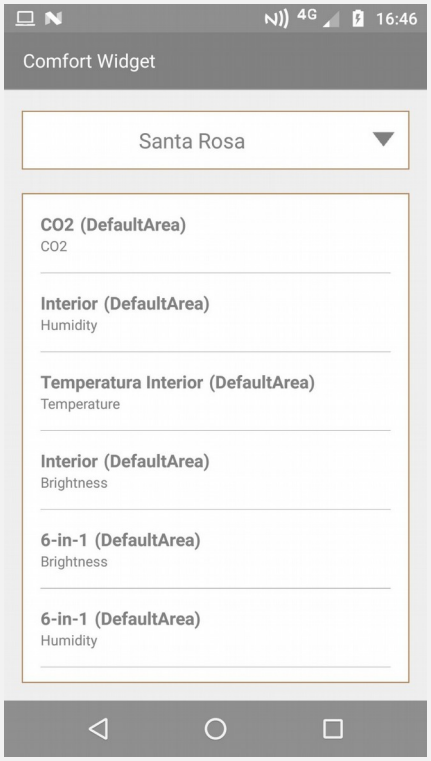
```
<receiver android:name=".widgets.ComfortWidget">
   <intent-filter>
      <action
android:name="android.appwidget.action.APPWIDGET_UPDATE" />
   </intent-filter>

   <meta-data
      android:name="android.appwidget.provider"
      android:resource="@xml/widget_comfort_info" />
</receiver>
```

- Activity

```
<activity
android:name=".widgets.ComfortWidgetConfigurationActivity"
   android:configChanges="orientation"
   android:label="Comfort Widget"
   android:screenOrientation="portrait">
   <intent-filter>
      <action
android:name="android.appwidget.action.APPWIDGET_CONFIGURE" /
>
   </intent-filter>
</activity>
```

Desarrollo

5. Actividad de configuración
   - Actividad encargada de configurar el *widget*
   - Spinner con apaptador personalizado
   - ListView con adaptador personalizado



Desarrollo

6. API

- Método para obtener exclusivamente sensores de confort

```
[Route("/installations/{Id}/sensors/comfort", Summary = "Gets a list of comfort sensors linked to the specified installation",
Verbs = "GET")]
    public class InstallationSensorsComfort : IReturn<InstallationSensorsComfortResponse>
    {
        [ApiMember(Name = "Id", Description = "Installation id", ParameterType = "path", DataType = "Guid", IsRequired = true)]
        public Guid Id { get; set; }
    }
```

```
public async Task<InstallationSensorsComfortResponse> Get(InstallationSensorsComfort request)
        {
            await RequestVerificationHub.VerifyRequest(request, this);
            return new InstallationSensorsComfortResponse
            {
                Sensors = await RetryAsync.Do(() => Db.SelectAsync<SensorWithAreaName>(Db.From<Installation>()
                        .Join<Floorplan>((i, f) => f.InstallationId == request.Id)
                        .Join<Floorplan, Area>((f, a) => f.Id == a.FloorplanId)
                        .Join<Area, AreaNode>((a, an) => a.Id == an.AreaId)
                        .Join<AreaNode, Node>((an, n) => an.NodeId == n.Id)
                        .Join<Node, NodeSensor>((n, ns) => n.Id == ns.NodeId)
                        .Join<NodeSensor, Sensor>((ns, s) => ns.SensorId == s.Id)
                        .Where<Sensor>(s => Sql.In(s.Type, SensorType.Brightness, SensorType.CO2, SensorType.Humidity,
SensorType.Thermostat, SensorType.MainThermostat, SensorType.Temperature, SensorType.ACIRTransmitter, SensorType.ThermostatRelay,
SensorType.Ultraviolet))
                        .Select("DISTINCT Sensor.*, Area.Name AreaName")))
            };
        }
```

Desarrollo

- Comunicación Aplicación - Servidor

```
@Route(Path="/installations/{Id}/sensors/comfort", Verbs="GET")
public static class InstallationSensorsComfort implements
IReturn<InstallationSensorsComfortResponse>
{
   @ApiMember(Name="Id", Description="Installation id",
ParameterType="path", DataType="Guid", IsRequired=true)
   public UUID Id = null;

   public UUID getId() { return Id; }
   public InstallationSensorsComfort setId(UUID value) { this.Id = value;
return this; }
   private static Object responseType =
InstallationSensorsComfortResponse.class;
   public Object getResponseType() { return responseType; }
}
```

```
public void getComfortSensors(String installationId, int retries,
SStackCustomCallback<InstallationSensorsComfortResponse>
callback) {
   InstallationSensorsComfort request = new
InstallationSensorsComfort();
   request.setId(UUID.fromString(installationId));

   executeRequest(request, HttpMethods.Get, retries, callback);
}
```

```
SStackWebApiConsumer.getInstance().getComfortSensors(installati
onId, 3, new
SStackCustomCallback<dto.InstallationSensorsComfortResponse
>() {
   @Override
   public void
onSuccess(dto.InstallationSensorsComfortResponse
responseObject) {
      sensorsList.clear();

      for(dto.SensorWithAreaName sensor :
responseObject.getSensors()) {
         sensorsList.add(new ViewModel(sensor.getId(),
sensor.getName(), sensor.getAreaName(), sensor.getType(),
sensor.getValue(), false, sensor.getBaseIconName(),
sensor.getUnitsForAverageGraph()));
      }

      adapter = new
ComfortWidgetSensorsListViewAdapter(sensorsList,
ComfortWidgetConfigurationActivity.this, mAppWidgetId,
installationName);
      sensorsListView.setAdapter(adapter);
   }

   @Override
   public void onFailure(Exception e) {

   }
});
```

Test

1. Pruebas de usabilidad
   - Llevadas a cabo por dos personas ajenas al desarrollo
   - Nexus 5 (Android 5.0), Moto 5 Plus (Android 7.0) y Huawei P10 (Android 7.0)
   - Resultado
     - Intuitivo y amigable
     - Tiempo requerido para concluir la actividad inferior a 20 segundos
     - Propuestas de mejoras
       - Actualización del nombre y area del sensor

Test

2. Pruebas unitarias
   - NUnit
   - Definición de una base de datos de prueba controlada

```
[Test]
public async Task GetComfortSensorsForInstallation()
{
    List<SensorWithAreaName> comfortSensors = (await installationService.Get(new InstallationSensorsComfort { Id
= testInstallation.Id })).Sensors;
    Assert.AreEqual(1, comfortSensors.Count());
    Assert.AreEqual(testSensor2, comfortSensors.Where(x => x.Id == testSensor2.Id.ToString()));
}
```

CONCLUSIONES

- Aprendizaje del funcionamiento de *widgets* dentro del sistema operativo Android
- Planificación del trabajo demasiado optimista
  - 4 desarrollos son demasiados para una primera vez
  - No se han planificado horas para la lectura y estudio de enControl
- Búsqueda de soluciones por encima de las limitaciones encontradas
- Mejora del servicio de datos en tiempo real