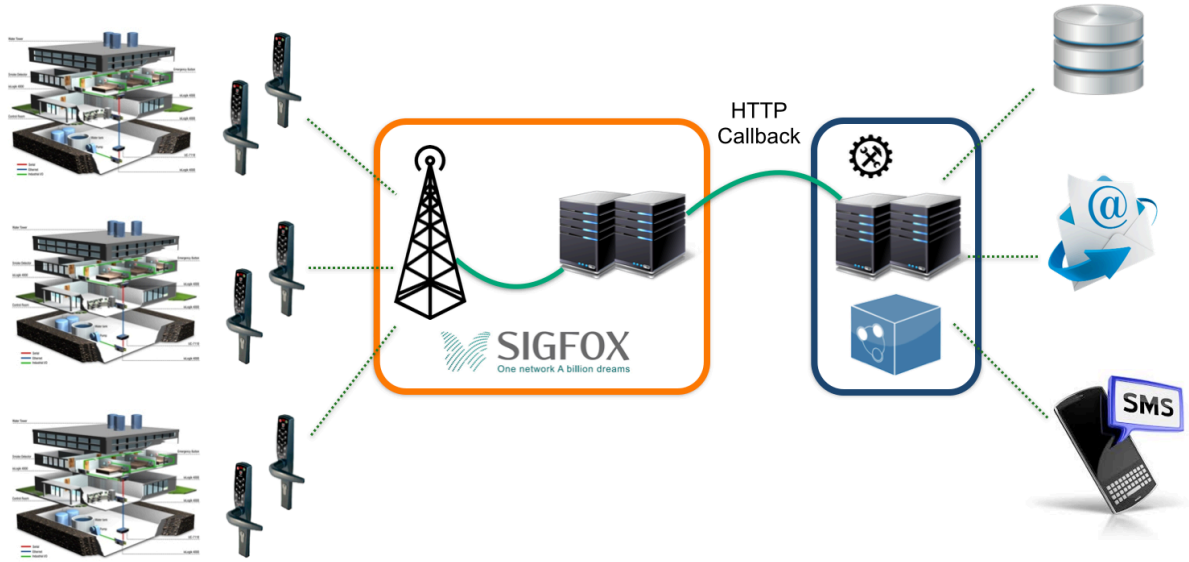
DoorLocker Manager

An M2M application dedicated to the management of connected door locks.

3 features to develop

- Real-time access logs
- Email notification for the battery state
- SMS notification in case of a forced lock



Sensors

DoorLocker (connected lock)

Locking / unlocking the lock

- Device sends true (lock) / false (unlock)

Lock state

- Device sends the state regularly (ex. every 10min).
- Possible lock states: forced, stuck, blackout, normal
- Possible battery states: low, normal, high
- new State("stuck,forced","20150101235959001", "low")

Actuators

Logger (database)

Sending an event: locking/unlocking

- new Event(“unlocked@user/location1", "20150101235959000")

Sending an event: lock state

- new Event("stuck,forced@user/location3", “20150101235959001")

Sending an event: battery state

- new Event("normal@user/location4", "20150101235959000")

Mailer (email service) / SMSSender (SMS service)

```
Contact admin = new Contact(“The Boss”, “demo1.diasuitebox@gmail.com",
                                    "+33652874356", "nomobile", “nourl", null);
List<File> files = new ArrayList<File>();
String content = … // battery value, lock state
```

Sending a message: Battery state

- sendMessage(admin, "Battery low”, content, files);

Sending a message: Forced lock

- sendMessage(admin, "DoorLock Alert”, content, files)

Taxonomy

```
device Device {
        attribute id as String;
        source isAlive as Boolean;
}

device PhysicalDevice extends Device {
        attribute location as String;
        attribute user as String;
}
---------------------------------- DOOR LOCKER ----------------------------------

device DoorLocker extends PhysicalDevice {
        action Lock;
        action UnLock;
        source locked as Boolean;
        source state as State;
}

structure State {
        state as String;
        timestamp as String;
        batteryLevel as String;
}
--------------------------------------- LOGGER ---------------------------------------

device Logger extends Service {
        action Log;
}

action Log {
        logEvent(event as Event);
}

structure Event {
        event as String;
        timestamp as String;
```

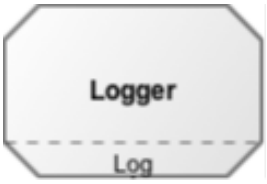
```
}
------------------------------------ MAILER ------------------------------------

device Mailer extends CommunicationService {}

device CommunicationService extends Service {
        action SendMessage;
}

action SendMessage {
        sendMessage(to as Contact, title as String, content as String,
                attachments as File []);
        sendMessageWithImage(to as Contact, title as String, content as
                        String, image as String);
}
---------------------------------- SMS SENDER ----------------------------------

device SMSSender extends CommunicationService {}

device CommunicationService extends Service {
        action SendMessage;
}

action SendMessage {
        sendMessage(to as Contact, title as String, content as String,
                attachments as File []);
        sendMessageWithImage(to as Contact, title as String, content as
                        String, image as String);
}
```

Development phases

1. Application design (30 min.)
2. Sending real-time access logs (implementation, test)
3. Sending a notification of low battery state via email (implementation, test)
4. Sending a SMS notification in case of a forced lock (implementation, test)

Instructions

- Indexing of context components is not needed to develop this application
- An Agenda device with addresses is not available, use the contact defined in the unitary test
- Multiple contracts of interaction can be defined in the context component
- Keep the code modular


Logger
Log


Mailer
SendMessage


SMS
Sender
SendMessage


locked
Door
Locker


state
Door
Locker