Laboratorul 07. Fitbit SDK - Clock Face

Până acum target-ul nostru a fost RaspberryPi, dar în acest laborator și următorul noul target va fi un produs Fitbit, și anume **Fitbit Versa 3** sau **Fitbit Sense** (produse lansate anul acesta). Vom utiliza *Fitbit Software Development Kit* (*SDK*) pentru a dezvolta o față de ceas (clock face) în cadrul acestui laborator, și o aplicație (app) în laboratorul următor, folosind mediul de dezvoltare pus la dispoziție de către Fitbit (Fitbit Studio) și simulatorul (Fitbit OS Simulator). Documentația pentru dezvoltatorii de aplicații Fitbit se poate găsi pe https://dev.fitbit.com/ [https://dev.fitbit.com/].

Setup

Setup-ul oficial poate fi găsit aici [https://dev.fitbit.com/getting-started/] la secțiunea *What You'll Need*.

Vă recomandăm să folosiți **Windows** sau **macOS** pentru aceste laboratoare!

Pentru a putea folosi infrastructura pusă la dispoziție de către Fitbit va trebui să urmați pașii:

1. Creați-vă un cont Fitbit accesând acest link [https://www.fitbit.com/signup].
2. Descărcați simulatorul (Fitbit OS Simulator) pentru Windows [https://simulator-updates.fitbit.com/download/latest/win] sau macOS [https://simulator-updates.fitbit.com/download/latest/mac]. Nu există support pentru Linux, dar există workarounds pentru acest lucru în cazul în care doriți să utilizați Linux la aceste laboratoare (urmați pașii de aici [https://github.com/bingtimren/fitbit-sim-starter] pentru a rula simulatorul pe Linux).
3. Accesați Fitbit Studio [https://studio.fitbit.com/], mediul de dezvoltare online pus la dispozție pentru developeri.

Conectați-vă atât la Fitbit Studio, cât și la simulator cu **același** cont Fitbit creat mai sus!

Arhitectura unei aplicații folosind Fitbit SDK

Informațiile următoare se regăsesc în documentația oficială:
https://dev.fitbit.com/build/guides/application/ [https://dev.fitbit.com/build/guides/application/]

O aplicație tipică are următoarea structură de directoare:

- `/app/` - directorul conține logica aplicației care urmează a fi executată pe device. Un fișier `index.js` trebuie să existe în acest director, altfel operația de build va eșua.
- `/companion/` - director opțional, conține logica companion-ului care este executată pe device-ul mobil (capabil să facă request-uri către Internet și să comunice cu aplicația). Dacă există un fișier `index.js` atunci componenta companion se va builda.
- `/common/` - director opțional, conține fișiere care pot fi folosite atât de aplicație, cât și de companion.
- `/settings/` - director opțional, conține informații despre setările aplicației.
- `/resources/` - directorul conține toate resursele care sunt folosite de aplicație.
- `package.json` - fișierul de configurare pentru aplicație.

Tipurile de resurse din directorul `/resources/` sunt:

- `/resources/index.view` - fișier obligatoriu, aici este definită interfața cu utilizatorul (fișier SVG).
- `/resources/widget.defs` - fișier obligatoriu, acesta controlează ce widget-uri de sistem sunt disponibile pentru a fi folosite în `index.view`.
- `/resources/*.css` - fișiere CSS care pot schimba aspectul elementelor din `index.view`.

- `/resources/*.png` și `/resources/*.jpg` - imagini ce pot fi folosite în `index.view` folosind tag-ul `<image>`.

Exerciții

În cadrul simulatorului setați **Versa 3** sau **Sense** ca de tip device (Settings → Device Type). Vom folosi versiunea 5.0 a SDK-ului.

0. Urmați primii 3 pași de la secțiunea *Create Your First Project* (Setup, Create a New Project, Installing) de pe https://dev.fitbit.com/getting-started/ [https://dev.fitbit.com/getting-started/]. În urma acestui exercițiu ar trebui să apară aceeași față de ceas (clock face) în simulator ca cea din ghid.

1. Ne propunem să creăm o nouă față de ceas (cea din GIF-ul de mai jos). Pentru aceasta o puteți edita pe cea de la exercițiul anterior sau să creați un nou proiect și să alegeți ca template tot *Digital Clock*. Descărcați template-ul pentru laborator de aici [https://drive.google.com/file/d/1DUrDt1szMPLCVOwq5MPfcUtjeSdRhygJ/view?usp=sharing] și aplicați-l în cadrul proiectului. Rulați aplicația în simulator.

Exercițiile din laborator se rezolvă în fișierul `app/index.js`. Template-ul conține comentarii de tip *TODO* pentru fiecare subpunct al exercițiilor. Vom utiliza doar Device APIs, documentația lor se află aici [https://dev.fitbit.com/build/reference/device-api/]. Limbajul de programare utilizat este Javascript, dar ne vom limita la funcționalitățile de bază.

2. Deocamdată fața de ceas indică doar ora, dar nu și statisticile afișate in cele 4 cadrane. Pentru acest exercițiu ne propunem să selectăm prima opțiune (pașii).

1. obțineți restul elementelor de UI (hr, floors, battery) și apoi adăugați acele variabile într-un array.

```
const arcHR = document.getElementById("hr");
const arcFloors = document.getElementById("floors");
const arcBattery = document.getElementById("battery");
const arcs = [arcSteps, arcHR, arcFloors, arcBattery];
```

2. creați un array pentru culorile celor 4 cadrane.

```
const arcColors = ["darkturquoise", "fb-red", "orange", "limegreen"];
```

3. completați funcția **initializeStat()** astfel incât să seteze ca opțiunea cu pași să fie cea selectată inițial (la start-up), funcția nu trebuie să întoarcă nimic. Trebuie să setați culoarea cadranului și a label-ului, dar și textul label-ului (`statLabel`). Inițializați variabila `currStat` care va stoca opțiunea curentă.

```
function initializeStat() {
  currStat = 0;
  arcs[0].style.fill = arcColors[0];
  statLabel.text = "Steps";
  statLabel.style.fill = arcColors[0];
}
```

Explicații:

- În Javascript, o constantă se definește astfel `const my_const = 0`.
- În Javascript, o variabilă se definește astfel `let my_var = 0`.
- În Javascript, un array se definește astfel `const cars = [“Volvo”, “BMW”, “Dacia”, “Skoda”]`.
- Pentru a obține un element UI putem folosi `document.getElementById(<id>)` din *Document* API, `<id>` fiind id-ul definit deja în `/resources/index.view`.
- Puteți alege orice culori doriți de aici [https://dev.fitbit.com/build/guides/user-interface/css/]. Cele utilizate în exemplu sunt “darkturquoise”, “fb-red”, “orange”, “limegreen”.
- Pentru a seta culoarea unui element x se poate folosi `x.style.fill = “red”`.
- Pentru a seta textul unui element text x se poate folosi `x.text = “Steps”`.

3. Ne dorim ca fața de ceas să itereze prin cele 4 cadrane la apăsarea ecranului (click pe ecran în simulator). Funcția care a fost setată să se apeleze la un eveniment de click/tap este funcția `changeStat()`. Completați-o astfel încât la fiecare apel să se selecteze următoarea opțiune (setând culoarea, textul îl vom seta la următorul exercițiu). Înainte de a seta noua opțiune asigurați-vă că ați deselectat-o pe cea anterioară (setați-i culoarea alb).

```
function changeStat() {
  arcs[currStat].style.fill = "white";
  currStat = (currStat + 1) % 4;

  arcs[currStat].style.fill = arcColors[currStat];

  statLabel.style.fill = arcColors[currStat];

  // this will change the stat label, don't remove
  fillInStatLabel();
}
```

4. Pentru ca fața de ceas să fie completă, trebuie să îi adăugăm și datele pentru fiecare opțiune selectată. Completați funcția `fillInStatLabel()` astfel încât la fiecare apel să se seteze text-ul afișat din variabila `statLabel` în funcție de opțiunea curentă. Va trebui să includeți și să apelați API-urile pentru *user-activity* (pași, floors), *heart-rate* (hr) și *power* (baterie) pentru a obține datele.

1. Importați API-urile de care avem nevoie pentru extragerea informațiilor din fiecare opțiune a cadranelor.

```
import { HeartRateSensor } from "heart-rate";
import { battery } from "power";
import { today } from "user-activity";
```

Modificați apoi în funcția `initializeStat()` textul de afișare a pașilor.

```
function initializeStat() {
  currStat = 0;
  arcs[0].style.fill = arcColors[0];
  statLabel.text = `${statNames[0]}: ${today.adjusted.steps}`; // this has been changed
  statLabel.style.fill = arcColors[0];
}
```

2. Completați funcția `fillInStatLabel()` astfel încât la fiecare apel să se seteze text-ul afișat din variabila `statLabel` în funcție de opțiunea curentă.

```
function fillInStatLabel() {
  switch(currStat) {
    case 0: // Steps
      statLabel.text = `${statNames[currStat]}: ${today.adjusted.steps}`;
      break;
    case 1: // HR
      if (HeartRateSensor && !hrm) {
        hrm = new HeartRateSensor({ frequency: 1 });
        hrm.start();
      }

      const hrmStr = hrm.heartRate == null ? '--' : `${hrm.heartRate}`;
      statLabel.text = `${statNames[currStat]}: ${hrmStr}`;

      break;
    case 2: // Floors
      statLabel.text = `${statNames[currStat]}: ${today.adjusted.elevationGain}`;
      break;
    case 3: // Battery
      statLabel.text = `${statNames[currStat]}: ${Math.floor(battery.chargeLevel)}%`;
      break;
  }
}
```

3. In `package.json` selectati permisiunile pentru *Activity* si *Heart Rate*.

Eplicații:

- Pentru a interoga variabila opțiunii curente `currStat` s-a utilizat un switch statement [https://www.w3schools.com/js/js_switch.asp].
- Textul afișat este de forma **<Opțiune>: <Valoare>**
- Pentru a crea un șir de caractere în care aveți mai multe variabile puteți folosi : `my_string = `${var_string}: ${var_number}``.

- Pentru a extrage numărul de pași și floors se utilizează variabila **today** din API-ul *user-activity* astfel: https://dev.fitbit.com/build/reference/device-api/user-activity/ [https://dev.fitbit.com/build/reference/device-api/user-activity/].
- Pentru a extrage procentajul bateriei se utilizează variabila **battery** din API-ul *power* astfel: https://dev.fitbit.com/build/reference/device-api/power/ [https://dev.fitbit.com/build/reference/device-api/power/].
- Pentru a extrage numărul de bătăi ale inimii pe minut se procedează astfel: https://dev.fitbit.com/build/guides/sensors/heart-rate/ [https://dev.fitbit.com/build/guides/sensors/heart-rate/]. **ATENȚIE**: folosiți variabila `hrm` pentru a nu inițializa de fiecare dată un nou obiect. Dacă dorim să ținem cont și de faptul că device-ul poate fi dat jos de pe mână puem folosi *Detecting Off-Wrist* din documentație: https://dev.fitbit.com/build/guides/sensors/heart-rate/#detecting-off-wrist [https://dev.fitbit.com/build/guides/sensors/heart-rate/#detecting-off-wrist]
- Permisiunile pentru *Activity* si *Heart Rate* permit utilizarea datelor utilizatorului (pași, floors, hr).

Resurse

- Template Laborator [https://drive.google.com/file/d/1DUrDt1szMPLCVOwq5MPfcUtjeSdRhygJ/view?usp=sharing]
- What is the file `package.json`? [https://nodejs.org/en/knowledge/getting-started/npm/what-is-the-file-package-json/]
- Getting Started with Fitbit SDK [https://dev.fitbit.com/getting-started/]
- Fitbit SDK - Device API Reference [https://dev.fitbit.com/build/reference/device-api/]
- Fitbit SDK Guides [https://dev.fitbit.com/build/guides/]

si/laboratoare/07.txt · Last modified: 2020/12/06 09:36 by stefan_radu.maftei