

---

---

# Intégration de statistiques en temps réel sur flux vidéo

*Real time statistics on live stream*

---

par

ZELLER QUENTIN

h e p i a

---

Haute école du paysage, d'ingénierie  
et d'architecture de Genève

Filière communications, multimédia et réseaux  
HAUTE ÉCOLE DU PAYSAGE, D'INGÉNIERIE ET D'ARCHITECTURE DE GENÈVE

Rapport concernant le travail de semestre du Bachelor en  
INGÉNIERIE DES TECHNOLOGIES DE L'INFORMATION dans la  
spécialisation COMMUNICATIONS, MULTIMÉDIA ET RÉSEAUX.

FÉVRIER 2018

Directeurs du travail : El Maliki Tewfiq, Revuelta Andres



## *Abstract*

A<sup>bstract</sup>



## *Remerciement*

J<sup>e</sup> remercie.



# *Table of Contents*

	<b>Page</b>
<b>Liste des tableaux</b>	<b>vii</b>
<b>Table des figures</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Cahier des charges de l'application . . . . .	1
1.1.1 État des lieux du marché . . . . .	2
<b>2 OpenCV &amp; Flask</b>	<b>5</b>
2.1 Fonctionnement général . . . . .	5
2.2 Environnement de développement de l'application test . . . . .	5
2.3 Environnement de production . . . . .	6
2.3.1 The Opera hybrid TV option . . . . .	6
<b>3 HBBTV</b>	<b>9</b>
3.1 Fonctionnement général . . . . .	10
3.2 Environnement de développement . . . . .	10
3.2.1 Opera TV Emulator . . . . .	10
3.3 Environnement de production . . . . .	10
3.3.1 The Opera hybrid TV option . . . . .	10
<b>A Annexe A</b>	<b>15</b>
<b>Bibliographie</b>	<b>17</b>



## *Liste des tableaux*

<b>TABLE</b>	<b>Page</b>
	vii



## *Table des figures*

<b>FIGURE</b>	<b>Page</b>
1.1 Schémas de fonctionnement applicatif . . . . .	2
2.1 The easy Flask website MJPEG. . . . .	7
2.2 Burn text on live stream, display on GUI . . . . .	8
2.3 The easy Flask website MJPEG. (code) . . . . .	8
3.1 HbbTV Broadband vs Broadcast . . . . .	9
3.2 Opera emulator user interface . . . . .	11
3.3 Opera emulator installation . . . . .	12
3.4 Opera emulator devlopper tools . . . . .	13



# *1 Introduction*

**C**e travail de recherche consiste en la reconnaissance des différents outils logiciels disponibles sur le marché, permettant l'ajout d'informations en temps réel sur un flux vidéo. Ceci dans le but d'apporter au client-consommateurs des données supplémentaires, notamment lors d'évènements sportifs. Nous utiliserons ici l'exemple du volleyball sans pour autant restreindre les recherches à ce seul but. Cette étude de cas concerne bien évidemment un domaine plus vaste que celui du volleyball ou même des jeux.

L'approche pratique de ce travail explorera aussi les différentes possibilités de collecter les informations statistiques via une interface fortement découpées ce qui permettra l'intégration des données de la manière la plus générique possible. Cette modularité permettra d'intégrer d'autre sources de données sans repenser l'intégralité de l'application et permettra une plus grande flexibilité quand au sport/jeux auquel elle s'adresse.

Nous commencerons par le recensement de toute les technologies différentes disponible sur le marché. Nous choisirons aussi le médium de communication le plus adéquats. De prime abord, il semblerait que deux groupes se distinguent de par les technologies qu'ils emploient. Le groupe télévisuel ordinaire qu'est la "télévision de salon" et un groupe plus orienté ordinateur/téléphone mobile. Nous étudierons s'il est possible de concilier ses deux groupes et choisirons le cas échéant le médium le plus adéquat.

La deuxième partie de ce travail consistera à différents tests sur les technologies ainsi que potentiellement quelques démonstrations de type "Hello World" afin de se rentre compte des ressources nécessaires à l'implémentations.

Si ce travail s'avère réalisable au regard des contraintes d'un mémoire de bachelor, il aboutira en l'implémentation d'une solution fonctionnelle.

## **1.1 Cahier des charges de l'application**

Le but final de ce travail est de trouver un moyen efficace, fiable et peut couteux de diffuser du contenu audio-visuel à valeur ajoutée et en temps réel. Plus concrètement il s'agit dans le cas d'une retransmission sportive de ; Premièrement récupérer le/les flux vidéos, les sources de données à incorporer aux flux vidéos. Puis de les combiner en un seul service multimédia soit par l'injonction directe des données dans le flux vidéo, ce qui reviendrait a "bruler" la vidéo avec les

informations ou alors, solution plus modulable, incorporer ces informations sur un flux parallèle dont nous discuterons au chapitre. 3. Le résultat final étant un flux audio-visuel ne nécessitant aucun applicatif supplémentaire pour fonctionner. Il devra être directement utilisable sur une télévision, un smart-phone ou un ordinateur et permettra donc la lecture sur n'importe quel appareil.

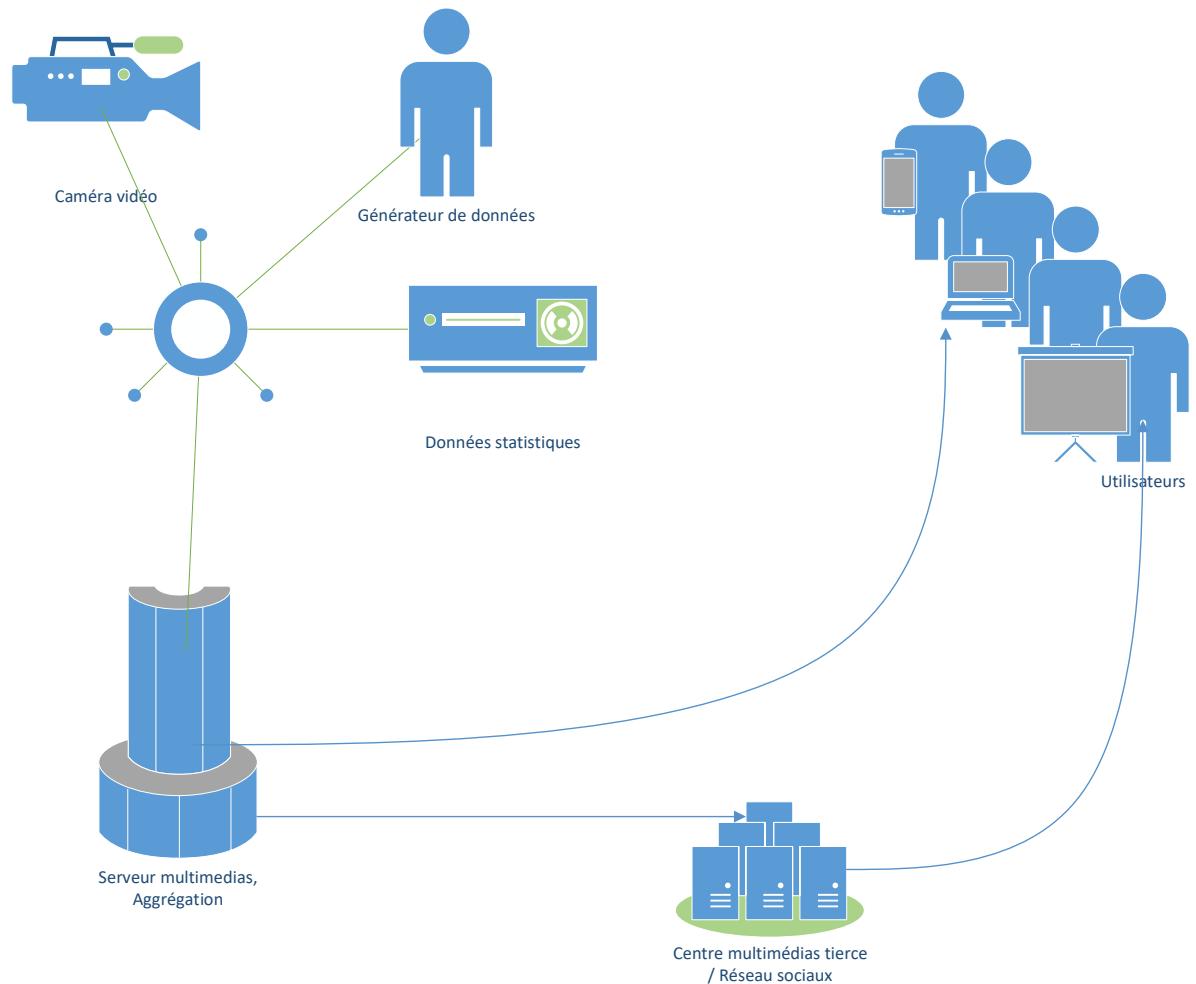


FIGURE 1.1 – Schémas de fonctionnement applicatif.

### 1.1.1 État des lieux du marché

Opencv ffmpeg wowsa + pluggin HBBTV -> opera

Application	Description	Difficulty
<b>OpenCV</b>	Une librairie BSD, init. Intel : Tout est à programmer sois-même. Très puissant car permet de traiter la vidéo image par image. Enorme librairie pour traitement d'image, etc... (C, C++, Python, Matlab...)	Difficile
<b>Wowza transcoder</b>	Librairie propriétaire, programmation objet (Java)	Moyen
<b>Wowza Clamp</b>	Un plugin RestFull propriétaire. Résultat très rapidement, Basé sur le principe d'overlay (Json), Prix : 300\$	Facile

### Wowza transcoder

### Wowza Clamp



## 2 OpenCV & Flask

OpenCV est une librairie multiplate-forme<sup>1</sup> et open source prévue pour le machine learning ainsi que pour le traitement d'image (computer vision)[8]. Du fait de sa licence BSD, cette librairie est constamment améliorée par le marché et notamment par les plus grosses entreprises du secteur technologique, ce qui en fait une des librairies leader dans ce domaine. En plus d'être disponible sur la quasi totalité des systèmes d'exploitations du marché elle est également disponible dans multiple langage que sont : C++, C, Python, Java and MATLAB à ce jour et sans compter évidemment les différents binding. Nous utiliserons ici la librairie pour Python qui est un langage certes moins optimisé que certains de ses concurrents mais qui à l'instar de Matlab permet de se concentrer sur le cœur du sujet. Il dispose en outre d'une bonne communauté et d'un bon support.

Une autre particularité d'OpenCV et ce qui le rend attractif à l'heure actuelle est sa compatibilité avec les processeurs Nvidia et leurs CUDA<sup>2</sup> Core[9]. Grâce aux avancées des GPUs à l'heure actuelle, principalement due au jeux vidéo mais aussi et principalement au crypto-monnaie les performances sont accrue d'un facteur 30x dans le pire des cas.

### 2.1 Fonctionnement général

OPENCV

### 2.2 Environnement de développement de l'application test

pycharm opencv, livrairies, flask...

```
# Usage:  
# 1. Install Python dependencies: cv2, flask. (wish that pip install works like a charm)  
# 2. Run "python main.py".  
# 3. Navigate the browser to the local webpage.  
from flask import Flask, render_template, Response  
from camera import VideoCamera  
  
app = Flask(__name__)
```

---

1. MacOS, IOS, Android, Windows, Linuxes,...  
2. CUDA : Technologie dite GPGPU (General-Purpose Computing on Graphics Processing Units)

```

@app.route('/')
def index():
    return render_template('index.html')

def gen(camera):
    while True:
        frame = camera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')
              

@app.route('/video_feed')
def video_feed():
    return Response(gen(VideoCamera()),
                   mimetype='multipart/x-mixed-replace; boundary=frame')

if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)
  
```

Listing 2.1 – Live streaming MJPEG with Flask, inspired from Miguel Grinberg. [10]

## 2.3 Environnement de production

### 2.3.1 The Opera hybrid TV option

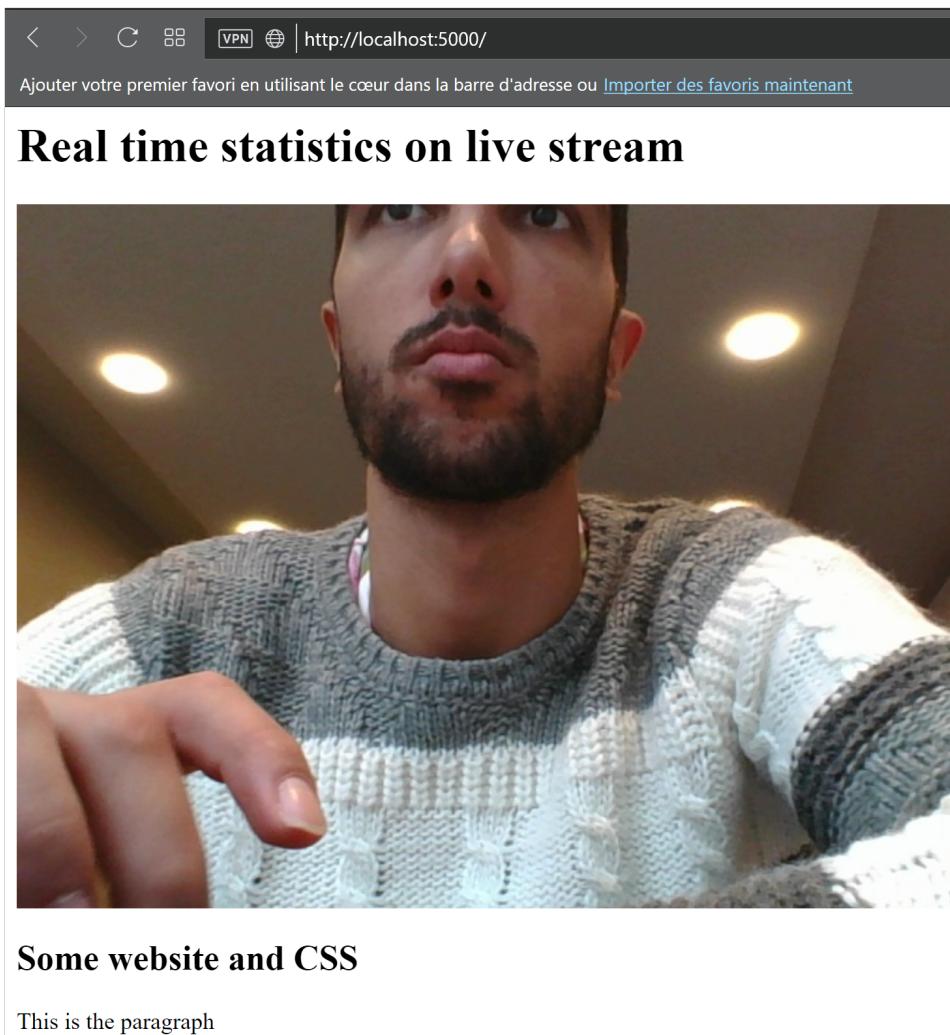


FIGURE 2.1. Streaming HTTP avec librairie python Flask

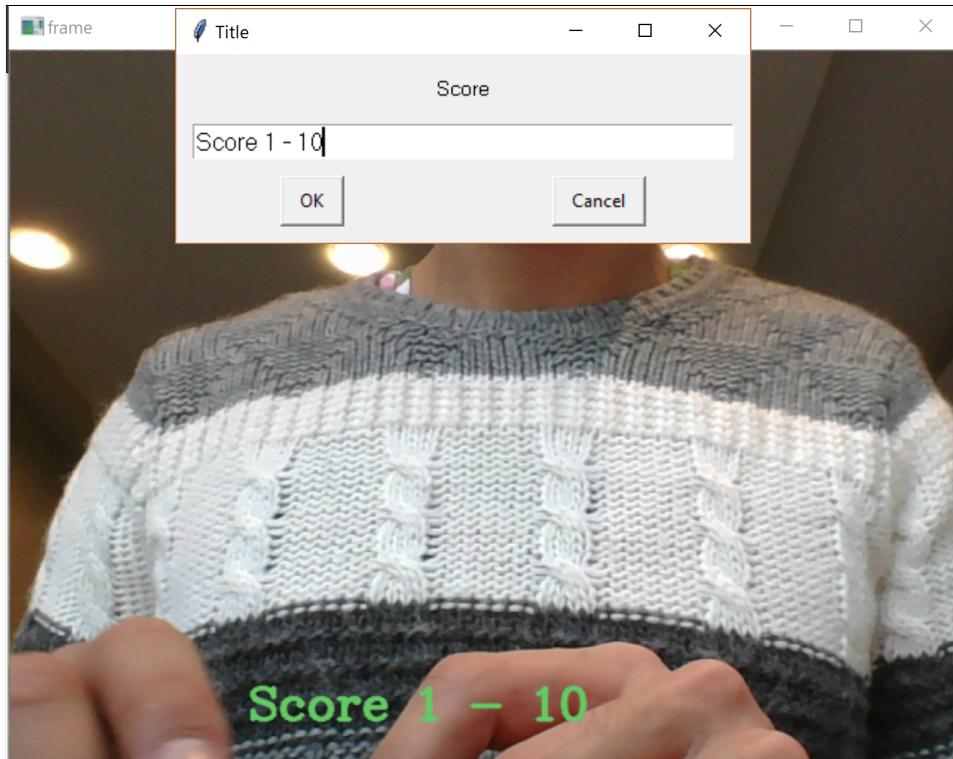


FIGURE 2.2. Intégration texte dans vidéo live, affichage GUI bureau

```

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

def gen(camera):
    while True:
        frame = camera.get_frame()
        yield (b"--frame\r\n"
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')

@app.route('/video_feed')
def video_feed():
    return Response(gen(VideoCamera()),
                   mimetype='multipart/x-mixed-replace; boundary=frame')

if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)

```

FIGURE 2.3. Code pour streaming HTTP avec librairie python Flask

### 3 HbbTV

**H**ybrid Broadcast Broadband TV, plus connu sous l'acronyme HbbTV est un standard européen permettant le partage d'information et de services en complément à un flux multimédias destinés à l'utilisateur final.

Il fait partie des outils ou protocoles TV OTT, acronyme signifiant télévision Over The Top ou services par contournement en français et qui définissent les contenus ne passant pas par le bouquet proposé par l'opérateur internet / télévision. C'est donc l'antithèse de la télévision linéaire, mode de consommation traditionnel. [1]

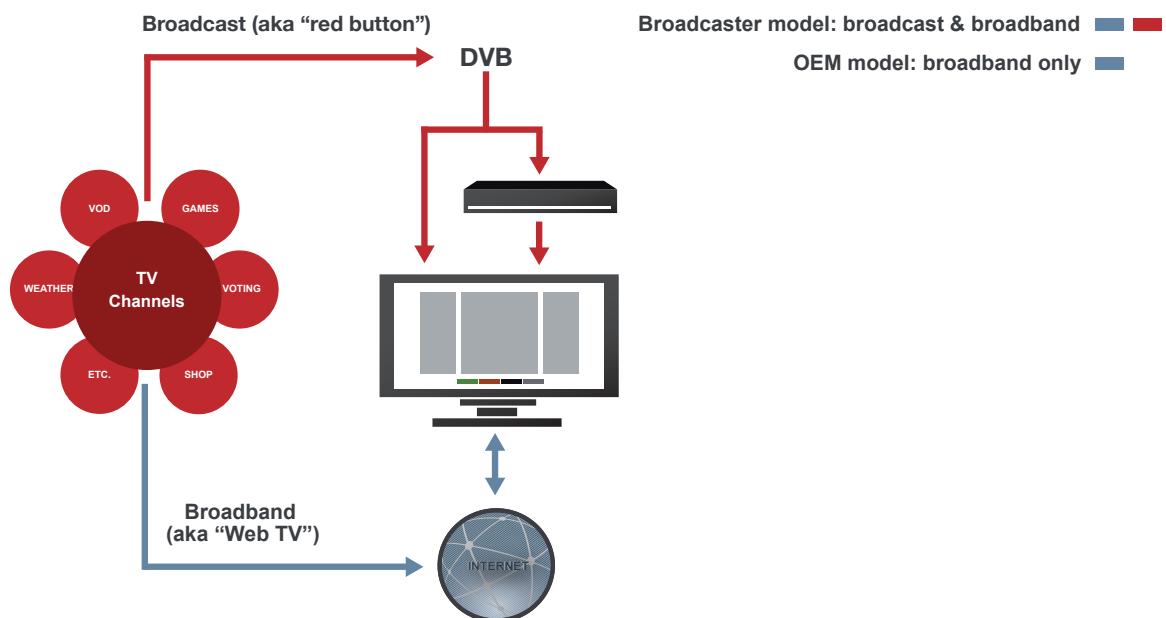


FIGURE 3.1 – HbbTV Broadband vs Broadcast

Le but du consortium est de créer un seul standard permettant d'acheminer du contenu broadcast et broadband à travers une seule interface de type web. Le but étant donc d'empêcher l'émergence de multiples standards propriétaires ainsi que les désavantages qui en incombe.[3] Les services proposés peuvent être infinis, passant de la vidéo à la demande, se dissociant donc du contenu vidéo de base jusqu'au quiz interactif lors d'une émission télévisée.

### 3.1 Fonctionnement général

### 3.2 Environnement de développement

Begins a section.

#### 3.2.1 Opera TV Emulator

L'émulateur Opera TV rend possible le développement de contenu HTML5 et CE-HTML pour différents appareils que sont ; Smart TV, lecteurs Blue-ray, Box et aussi les ordinateurs.[?] La position sur le marché du leader Opera rend quasi indispensable cet environnement de développement, du moins pour les tests.

L'environnement de développement se caractérise dans une machine virtuelle tournant sur le bien connu, VirtualBox. Il permet de s'abstraire de l'accès physique à une machine/TV ainsi que de rendre plus prédictif les protocoles de tests. Il dispose de deux interfaces graphique, la première étant le flux vidéo de la sortie standard qui propose une interface "TV-like" et dont la sortie sera celle de notre application développée. Sur la figure [3.2(e)] ci-après nous voyons la "landingpage" de l'émulateur opera. Celui-ci nous permet d'entrer le lien du l'application HbbTV que nous souhaitons grâce à un clavier virtuel. Cependant cet interface n'est pas très ergonomique et gère mal l'interaction clavier-souris.

Une autre interface beaucoup plus efficace nous est proposée. Il s'agit de l'interface web, disponible par défaut sur le port 5555. Elle dispose notamment de fonctions utiles au debugging (fps counter, paint regions, terminal event, javascript event, performance monitor...) [3.3], et un certain nombre paramètre de fonctionnement de l'émulateur [3.2(a) 3.2(b)] et du SDK ainsi que les différentes compatibilités assurées. Cette interface propose aussi une télécommande virtuelle [3.2(f)] nous permettant de tester les différentes réactions au événements boutons.

### 3.3 Environnement de production

#### 3.3.1 The Opera hybrid TV option

The Opera hybrid TV option est basée sur leur propre Opera Devices SDK (Standard Developpement Kit) qui est une technologie déjà bien répandue sur le marché avec plusieurs millions de set-up box dans le monde. Ce kit de développement implémente plusieurs module de compatibilité tel que (OIPF, CE-HTML, CEA-2014...) et de ce fait supporte toutes les applications HbbTV. Ils se targuent aussi d'avoir une technologie éprouvée permettant des performances et intégration supérieure à ce que propose le marché.[3]

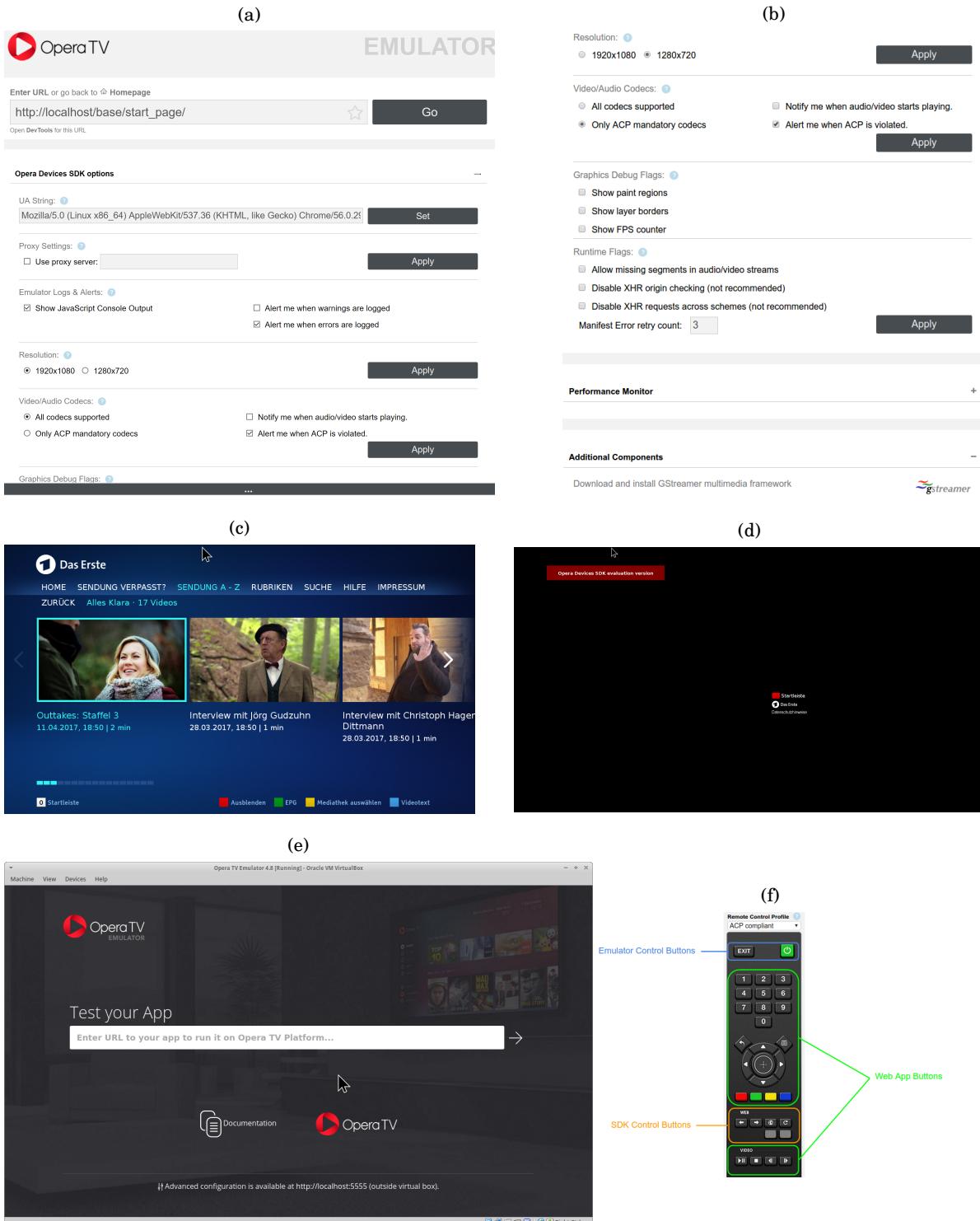
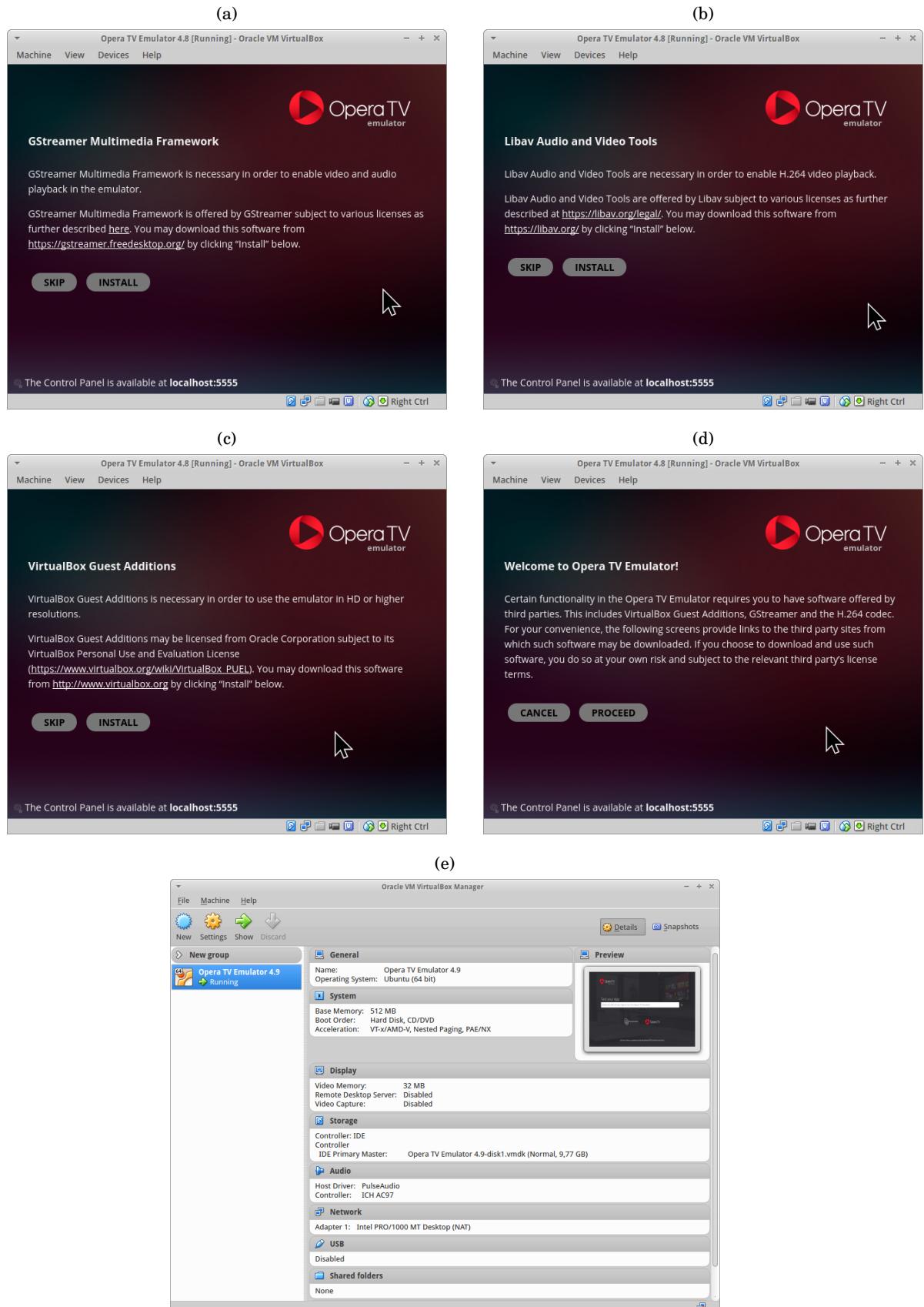
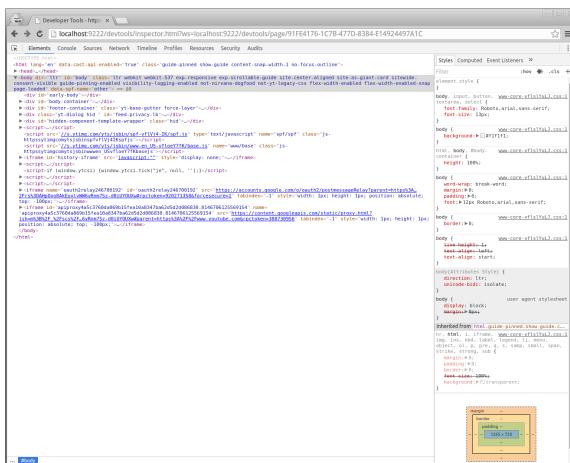


FIGURE 3.2. Interface utilisateur.



(a)

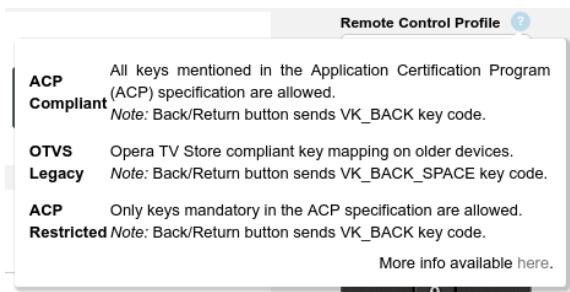


(b)

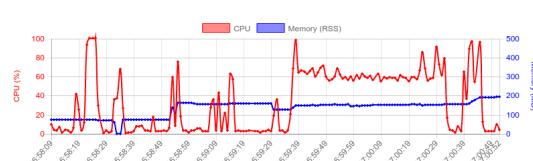
Key Mappings

Key Name	ACP Compliant Mapping	OTVS Legacy Mapping	ACP Restricted Mapping
←	VK_LEFT	VK_LEFT	VK_LEFT
→	VK_RIGHT	VK_RIGHT	VK_RIGHT
↑	VK_UP	VK_UP	VK_UP
↓	VK_DOWN	VK_DOWN	VK_DOWN
Confirm>Select/OK	VK_ENTER	VK_ENTER	VK_ENTER
Back/Return	VK_BACK	VK_BACK_SPACE	VK_BACK
Menu	VK_MENU	VK_MENU	
RED	VK_RED	VK_RED	
GREEN	VK_GREEN	VK_GREEN	
YELLOW	VK_YELLOW	VK_YELLOW	
BLUE	VK_BLUE	VK_BLUE	
0	VK_0	VK_0	
1	VK_1	VK_1	
2	VK_2	VK_2	
3	VK_3	VK_3	
4	VK_4	VK_4	
5	VK_5	VK_5	
6	VK_6	VK_6	
7	VK_7	VK_7	
8	VK_8	VK_8	
9	VK_9	VK_9	
PLAY/PAUSE			
STOP	VK_STOP	VK_STOP	
REWIND	VK_REWIND	VK_REWIND	
FORWARD	VK_FAST_FWD	VK_FAST_FWD	

(c)



(d)



(e)

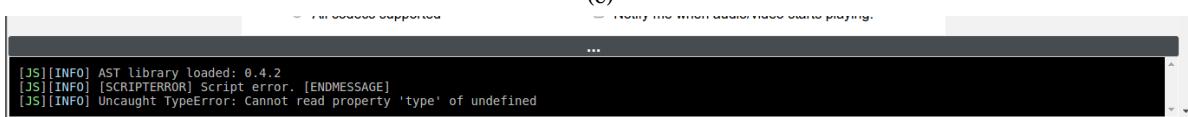


FIGURE 3.4. Outils developpeur.



## *A Annexe A*

**A**<sup>nnexe</sup>



## *Bibliographie*

- [1] WIKIPEDIA, *Over-the-top media services*, [https://en.wikipedia.org/wiki/Over-the-top\\_media\\_services](https://en.wikipedia.org/wiki/Over-the-top_media_services), 28 January 2018, at 13 :23.
- [2] OPERA PRESS, *Deliver seamless entertainment experiences*, The Opera hybrid TV option, 1 February 2018., [www.opera.com/media/b2b/tv/Opera-TV-Emulator.pdf](http://www.opera.com/media/b2b/tv/Opera-TV-Emulator.pdf), pp. 1.
- [3] OPERA PRESS, *Opera TV Emulator*, The Opera TV emulator introduction, 2 February 2018., pp. 1.
- [4] s J. D. MASUCCI AND J. W. SCHIEFELBEIN, *The rhd6 mutation of arabidopsis thaliana alters root-hair initiation through an auxin- and ethylene-associated process*, Plant. Physiol., 106 (1994), pp. 1335–1346.
- [5] R. PAYNE AND C. GRIERSON, *A theoretical model for rop localisation by auxin in arabidopsis root hair cells*, PLoS ONE, 4 (2009), p. e8337. doi :10.1371/journal.pone.0008337.
- [6] S. RIGAS, G. DEBROSSES, K. HARALAMPIDIS, F. VICENTE-ANGULO, K. A. FELDMAN, A. GRABOV, L. DOLAN, AND P. HATZPOULOS, *Trh1 encodes a potassium transporter required for tip growth in arabidopsis root hairs*, The Plant Cell, 13 (2001), pp. 139–151.
- [7] HBBTV APPLICATION DAS ERSTE ,*Example of application with video content* , <http://hbbtv.daserste.de/index.php>, Main application, 30 January 2018.
- [8] OPENCV OFFICIAL ,*About OpenCV, Support Question* , <https://opencv.org/about.html>, <https://opencv.org/platforms/>, Open Source Computer Vision Library, 4 February 2018.
- [9] OPENCV OFFICIAL ,*About OpenCV, CUDA support* , <https://opencv.org/platforms/cuda.html>, Performances, 4 February 2018.
- [10] MIGUEL GRINBERG ,*Video Streaming with Flask* , <https://blog.miguelgrinberg.com/post/video-streaming-with-flask>, Code, October 20 2014.

