Lockatme
A screen lock with facial recognition abilites

David Anandanadaradja, Sagar Gueye, Bruno Inec,
Matthieu Kirschleger, Pierre-Louis Sergent

16 janvier 2018

Table des matières

Chapitre 1

Présentation des membres du groupe

1.1 Contexte

Dans le cadre de notre DUT Informatique à l'IUT Lyon 1, nous sommes tenus de réaliser un projet tuteuré durant le second semestre. Ce projet s'étendant également sur le troisième semestre, il a pour but de répondre à une problématique précise et de mettre en oeuvre les compétences acquises au cours de la formation. Il a aussi vocation à faire découvrir de nouveaux domaines et il nous permettra d'élargir nos savoirs à travers une auto-formation.

Ce projet se découpe en deux axes :
— Rédaction cahier des charges (second semestre)
— Réalisation du projet en lui même (troisième semestre)

Malgré une liste de sujets proposés, notre groupe a voulu suivre ses propres motivations (présentées plus loin dans ce document) et a choisi de proposer un sujet à M. Vidal. L'intitulé est le suivant : Verrouillage et déverrouillage d'écran par reconnaissance faciale sous Linux.

1.2 Organisation et membres

L'équipe chargée de ce projet est constituée
— Tuteur du projet : M. Vincent VIDAL
— Chef du projet : M. Bruno INEC
— Membres : M. David ANANDANADARADJA, Mme Sagar GUEYE, M. Matthieu KIRSCHLEGER et M. Pierre-Louis SERGENT

1.3 Compétences

Notre projet comporte deux contraintes principales : il nécessite une bonne connaissance du langage Python (explication choix outils cf III) et une maîtrise de Linux. L'impulsion de ces choix vient en grande partie du chef de projet qui possède une expérience importante dans ces deux domaines. David et Pierre-Louis possèdent quant à eux une expérience modérée dans l'utilisation de Linux (distribution Arch). L'ensemble des compétences individuelles est résumé ci après :

Python :
— Confirmé : Bruno INEC
— Intermédiaire : Pierre-Louis SERGENT
— Débutant : Sagar GUEYE, Matthieu KIRSCHLEGER, David ANANDA
Linux :
— Confirmé : Bruno INEC
— Intermédiaire : David ANANDA, Pierre-Louis SERGENT
— Débutant : Sagar GUEYE, Matthieu KIRSCHLEGER

Comme le montre le listing précédent, les compétences du groupe sont très disparates. Cela peut apparaître comme une contrainte, mais en réalité cela constitue une véritable opportunité pour tous les membres. Ils vont ainsi pouvoir se former dans les domaines ci après. Ils sont essentiels pour la suite des études et pour le milieu professionnel.

— Programmation : Linux, Python, TeX
— Rédaction cahier des charges
— Travail en équipe : réunion, communication, CI, modèle de développement

Nous étions donc motivés pour nous lancer dans un sujet avec nombre d'inconnus mais qui allait être fort enrichissant.

À noter également que dans un projet s'étendant sur une telle durée les compétences humaines et plus généralement les compétences annexes à l'informatique ne sont pas à négliger.

Chapitre 2

Présentation du Projet

2.1 Buts

Le but premier de l'application est de déverrouiller un écran d'ordinateur, à l'aide d'une caméra, par reconnaissance faciale. Cependant cela implique de mettre en place un verrouillage d'écran sous Linux. Les URS spécifiques seront décrit plus tard dans ce document.

2.2 Motivations

Trois membres du groupe utilisent Arch Linux qui est une distribution minimale de Linux. Le fait de quitter Windows leur à permis de pleinement se concentrer sur la machine à un plus bas niveau, avec tous les avantages de liberté qu'offre une plateforme open source, mais aussi toutes les contraintes qui sont très formatrices et qui forcent à se pencher d'avantage sur le fonctionnenement de ce système d'exploitation. Les trois utilisateurs cherchaient une manière de verrouiller/déverrouiller leur écran de manière sécurisée. Et l'idée de ce projet a fleuri suite à un article présent dans le magazine Linux Magazine/France n°203 : "Mettez en place un système de reconnaissance faciale".

2.3 Linux

Le développement du logiciel se fera sur Linux. Un tel projet sur Windows aurait été bien plus difficile concernant l'implémentation système mais aussi le code de l'application. De plus, l'OS est largement privilégié par les développeurs dans le monde de la programmation. C'est pourquoi nous avons choisi de réaliser notre projet sous Linux, qui s'adressera donc à un public famillié avec la CLI (Command Line Interface) et les autres aspects techniques. Des interfaces seront potentiellement développées à terme pour les utilisateurs de distributions plus user-friendly (comme Ubuntu).

2.4 Open Source

Le développement du projet se fera de manière complètement transparente et donc en open source. Ce choix est assez logique lorsque l'on réalise un programme pour Linux, car il s'inscrit exactement dans la politique des développeurs qui ont réalisé ce dernier. Cela possède de nombreux avantages : possibilité pour la communauté de contribuer au projet au travers de modifications du code, commentaires, rapport de bug, ...

Chapitre 3

Stratégie de développement

3.1 Méthologie de développement

3.1.1 Définition

Ces derniers temps, un nouveau groupe de méthodes fait son apparition dans la gestion de projet : on parle de méthodes agiles et Scrum fait partie de celles-ci (voir figure 3.1). Le terme "agile" définit une approche de gestion de projet qui prend le contre-pied des approches traditionnelles prédictives et séquentielles de type cycle en V ou waterfall. La méthode dite « traditionnelle » favorise l'élaboration d'un plan détaillé du besoin du client laissant donc peu de place au changement. Cela entraîne souvent un effet tunnel, c'est-à-dire un manque de communication entre le maître d'œuvre et le maître d'ouvrage, qui peut être néfaste pour mener à bien le projet. Les membres du groupe de travail se rapportent alors aux spécifications validées et au contrat. Certains projets se terminent dans la douleur au risque de compromettre la relation client. L'approche agile, elle au contraire, permet de réduire considérablement cet effet tunnel en donnant davantage de visibilité. Pour cela, on implique le client du début à la fin du projet et on adopte un processus itératif et incrémental (processus qui implique l'intégration continue de l'architecture d'un système pour produire des versions exécutables, chaque nouvelle version contenant des améliorations incrémentales).

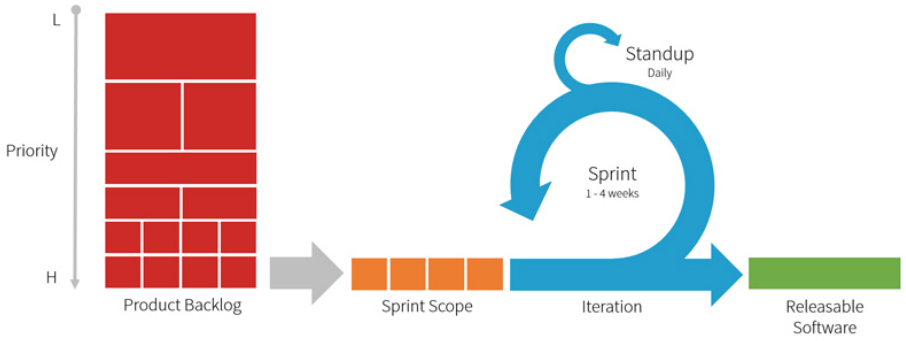

FIGURE 3.1 – schéma agile et scrum

3.1.2 Application dans le projet lockatme

Concernant notre projet, nous avons décidé d'opter pour une stratégie agile à défaut d'une planification détaillée. Il faut d'abord fixer un premier objectif à court terme (rédaction d'une partie du cahier des charges par exemple). Dès que ce premier objectif est atteint, on adapte le plan de réalisation en fonction

de ce qui a été fait et pas fait. Ce processus se répète jusqu'à la version finale du produit. Dans notre cas - l'élaboration d'un projet de développement logiciel - nous avons élaboré une vision du produit puis la liste des exigences que cela implique : c'est la phase d'itération. Chaque itération comprend des travaux de développement, test et/ou conception. A la fin de chaque itération, les membres du groupe présentent les produits intermédiaires. C'est à ce moment qu'on peut donner des feedbacks qui seront importants sur les itérations suivantes. Cette méthode, en plus de son efficacité, apporte de la confiance entre les différents membres. On peut par la suite, modifier les priorités des fonctionnalités qui n'ont pas encore été ajoutées au produit. Grâce à l'approche agile, nous avons donc une véritable souplesse dans la réalisation du produit permettant un travail efficace et un solide esprit d'équipe.

3.2 Intégration continue

Afin de collaborer au mieux au sein de l'équipe, nous avons mis en place une C.I. (voir figure 3.2). C'est un pipeline qui consiste à executer les tests mis en places automatiquement après chaque changement sur une branche du projet.

Après avoir positivement passé les tests et avoir subit une phase de review pour recevoir commentaires et conseils, le code peut être intégré dans la branche principale avec l'assurance (relative à l'exhaustivité des tests) qu'il n'a pas compromis les autres parties du code.

Cette méthodologie de travail permet à l'utilisateur final de récuperer toutes les nouveautés introduites au moment où elles sont *merged* et non à la sortie d'une nouvelle version. Dans certain cas, notamment avec les sites web par exemple, on peut même arriver à faire du Continuous Deployment, ce qui veut dire qu'il n'y a plus besoin de versionner les différents états du projet et qu'à chaque push d'un des développeur, le site disponible à l'utilisateur est celui contenant le dernier commit. Ceci n'est pas possible à faire pour un projet comme le notre, qui doit être manuellement mis-à-jour à chaque nouvelle version.

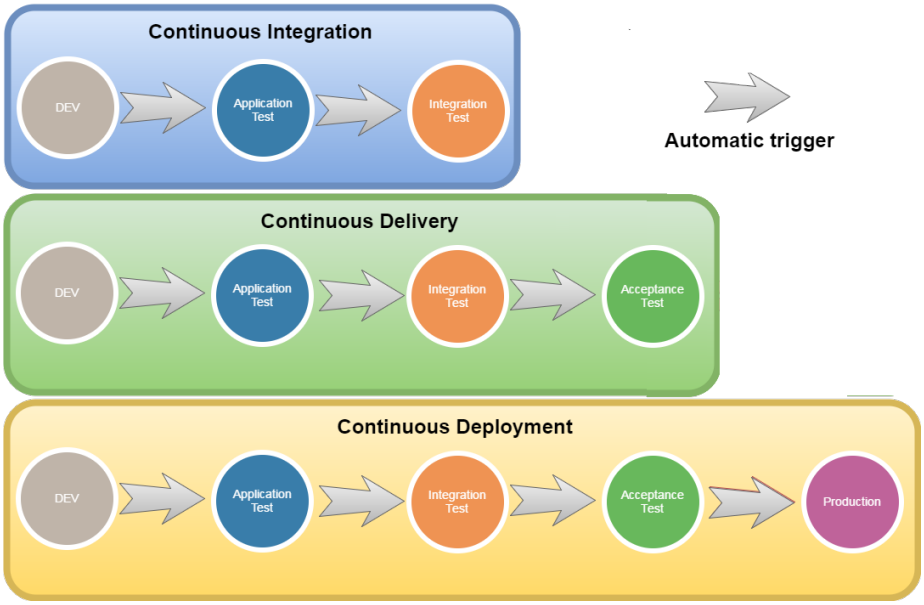

FIGURE 3.2 – continious integration

Chapitre 4

Phase du projet – Distribution des tâches

4.1 Contexte

Le projet repose, en assez grande partie, sur des domaines qu'il nous fallait découvrir et étudier. C'est pourquoi l'attribution et la définition des tâches à été une problématique importante dans le projet lockatme. Cette étape est essentielle pour le bon déroulement du développement. Nous nous sommes efforcés d'attribuer les tâches à des duos ou trios afin de permettre deux choses : une auto-formation étant parfois nécessaire, les membres assignés à une même tâche pouvaient partager les connaissances et les difficultés, afin de faire avancer le développement plus vite. De plus les membres ne devraient pas rester bloquer sur une tâche étant donné que les attributions se font en fonction des compétences de chacuns.

4.2 Listes des tâches

La liste des tâches est fortement susceptible d'évoluer au cours du projet en fonction de l'évolution de celui-ci, des difficultés recontrées et du temps imparti. La réalisation des tests n'est pas spécifiée dans cette liste. Niveaux d'importances : 1 - essentiel, 2 - important, 3 - importance modérée, 4 - optionnel.

T1 : Cahier des charges(1)
— conception
— rédaction
— validation
T2 : Déverrouillage
— algo reconnaissance faciale(1)
— algo déverrouillage par mot de passe(1)
— gestion des cas d'erreurs(1)
— fichier configuration/personnalisation(2)
T3 : Implementation système, installation
— verrouillage écran(1)
— déverrouillage écran(1)
— installation qualification(2)
— upload packages sur pip/yaourt(3)
— readme(2)
T4 : Amélioration
— interface graphique(3)
— développemement plateforme commune de verrouillage/déverrouillage d'écran(4)

4.3 Repartition des tâches

Du fait de la disparité des compétences du groupe, il était essentiel d'associer plusieurs personnes à chaque tâche. Afin de faciliter le travail mais aussi pour que chacuns apprennent de nouvelles choses. Comme dis plus haut nous avons

donc décider d'attribuer chaque sous-tâche à des duos ou trios ; composés d'une personne référente et d'une personne peu ou pas expérimenté dans le domaine ciblé. Ainsi la personne ayant plus de connaissance est capable de diviser le travail en d'autres sous-tâches (non présente sur le diagramme) afin de faire participer au mieux les autres membres.

Nous avons prit soin d'associer les tâches bloquantes aux personnes compétentes. De plus si une difficulté survient sur une tâche, les réunions hebdomadaires permettent de faire le point et d'aider la personne.

4.4 Diagramme de Gantt

Voici le diagramme de Gantt résumant les principales tâches et sous-tâches ainsi que la repartition de celles-ci.

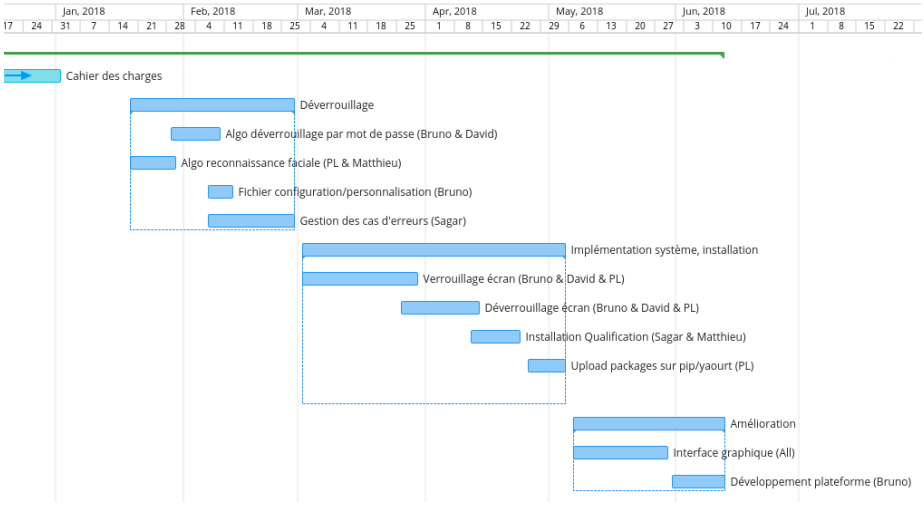

FIGURE 4.1 – diagramme de Gantt

Chapitre 5

User Requirement Specifications

5.1 Définition

Les *User Requirement Specifications* (URS), ou le cahier des charges des spécifications de l'utilisateur, à pour but de présenter les exigences du client ou de l'utilisateur. Il doit contenir une liste exhaustive (dans la mesure du possible) de ce que le logiciel sera en mesure de faire au terme de son développement. Dans notre cas, les exigences sont tout d'abords posé par nous, premiers utilisateurs. À terme, nous espérons élargir cela à une communauté se formant autour du programme.

5.2 Priorité

Nous avons décidé de diviser les URS en trois niveaux de priorité afin de bien séparer les caractéristiques que nous voulons absoluement trouver dans le logiciel final et les caractéristiques qui pourraient être intéressantes à développer dans le futur.

- M – Mandatory requirement. **doit être développé** 'Cette caractéristique doit être incluse dans le logiciel final.'
- D – Desirable requirement. **doit être développé dans la mesure du possible** 'Cette caractéristique devrait être incluse dans le logiciel final, sauf si contrainte trop importante ou manque de temps.'
- E – Possible future enhancement. **peut être développé dans le futur ou sera développé en fonction de l'avancement du projet** 'Cette caractéristique pourrait être incluse dans le logiciel final dans le futur.'

5.3 Mandatory requirement

5.3.1 Explication générale(1)

Le logiciel lockatme devra être installable sur Linux et aura pour fonctionalité de permettre à l'utilisateur de vérrouiller son écran puis de le déverrouiller par reconnaissance faciale ou, le cas échéant, par mot de passe. Pour se faire l'utilisateur sera invité à enregistrer des photos de lui dans un dossier spécifique, ces photos seront utilisé comme modèle pour la reconnaissance par webcam. Des tests seront effectué pour déterminer si ces photos sont de qualités suffisantes pour reconnaître l'utilisateur à coup sûr. Le paramétrage du programme se fera à travers un fichier qui correspond aux normes actuelles des fichiers de configuration.

5.3.2 Organisation(1)

- M.1 : Verrouillage écran
  L'utilisateur sera capable de verrouiller son écran avec une commande.

— M.2 : Reconnaissance faciale
Le logiciel sera capable de reconnaître le visage de l'utilisateur à l'aide d'une webcam.

— M.3 : Déverrouillage écran
L'utilisateur sera capable de déverrouiller son écran.
  — M.3.1 Par reconnaissance faciale
    — M.3.1.1 Prise d'information
      Le logiciel sera capable de prendre une photo toutes les trois secondes à travers la webcam. Le cas échéant cf M.3.2

  — M.3.2 Par saisie de mot de passe
    En cas d'échec de la reconnaissance faciale, l'utilisateur sera rédirigé vers un déverrouillage d'écran par mot de passe.

5.4 Desirable requirement

5.4.1 Explication générale(2)

De plus nous souhaiterions que l'utilisateur puisse personnaliser plusieurs aspects du logiciel. Par exemple, il lui serait possible de choisir une image lors du verrouillage de l'écran. Il pourrait également modifier la fréquence de prise d'information lors du déverrouillage par reconnaissance faciale. Concernant le verrouillage de l'écran, nous souhaiterions qu'il se fasse de manière automatique lorsque l'utilisateur baisse le clapet, ou qu'il appuie sur une touche choisie. Si nous avons de le temps nous souhaiterions réaliser une interface graphique pour le choix des images «modèles» et pour l'entrée du mot de passe.

5.4.2 Organisation(2)

— M.1 Verrouillage écran
  — D.1 Verrouillage simplifié de l'écran
    L'utilisateur devrait être capable de verrouiller son écran avec :
    — D.1.1 Un raccourci clavier *
    — D.1.2 La fermeture clapet *

  — D.2 Personnalisation de l'écran verrouillé
    L'utilisateur devrait être capable de choisir l'aspect de son écran verrouillé (image, texte).

— M.3.1.1 Prise d'information
  — D.3 Choix fréquence prise photo
    L'utilisateur devrait être capable de choisir la fréquence de prise de photos lors du déverrouillage par reconnaissance faciale.

— D.4 Interface graphique
L'utilisateur devrait être capable de choisir ses photos et entrer son mot de passe lors du déverrouillage grâce à une interface graphique.

* : Il est important de rappeler que le logiciel sera disponible uniquement sur Linux. L'utilisateur sera donc très libre pour les configurations, et l'utilisation d'un raccourci clavier ou du verrouillage automatique par fermeture du clapet ne pourra se faire qu'à travers des changements réalisés par l'utilisateur. La marche a suivre sera indiqué dans le mode d'emploi. Mais l'utilisateur restera très libre de ses actions.

5.5 Possible future enhancement

5.5.1 Explication générale(3)

Le projet étant open source nous invitons les développeurs à consulter le repository GitHub pour contribuer au projet et proposer de nouvelles fonctionnalités. L'idée du projet est que lockatme pourra être utilisé comme programme de base pour développer de nouveaux moyens de déverrouillage d'écran. Nous avons notamment pensé à un déverrouillage par finger-print (pour les ordinateurs équipés). Ainsi l'utilisateur aurait la possibilité de choisir parmis les différents moyen de déverrouillage pour sécuriser son ordinateur. Une autre avancée souhaitable serait la récupération de photos des réseaux sociaux (via leurs API respectives) afin que l'utilisateur n'ait pas à choisir des photos manuellement. Enfin pour garantir la sécurité du déverrouillage, il serait souhaitable d'améliorer la reconnaissance faciale, pour faire face à de possibles failles (montrer une photo de l'utilisateur à la webcam).

5.5.2 Organisation(3)

— M.3 Déverrouillage écran
  — E.1 Programme modulaire
    — E.1.1 Par Finger-print
      L'utilisateur pourrait être capable déverrouiller son écran avec un finger-print.

— M.2 Reconnaissance faciale
  — E.2 Récupération de photos automatique
    L'utilisateur pourrait être capable d'entrer son nom de compte et mot de passe d'un réseau social qui permet au travers de son API d'utiliser les photos de l'utilisateurs ainsi que les metadatas sur l'identité des personnes présentes sur celles-ci.

— M.3.1 Déverrouillage écran par reconnaissance faciale
  — E.3 Instruction spécifique

Le programme pourrait être capable de demander à l'utilisateur de faire certaines gestuelles spécifiques du visage.

Chapitre 6

Software Design Specifications

6.1 Définition

Les *Software Design Specifications* (SDS), ou le cahier des charges de conception du logiciel, à pour but de présenter l'architecture générale du logiciel. La présentation se fait à deux échelles : à bas niveau avec des exemples de codes et l'explication des bibliothèques utilisées. Et à un plus haut niveau avec des diagrammes de fonctionnement, des exemples d'utilisations et des captures d'écran. À noter que c'est dans cette section que nous traiterons des limites du logiciel. C'est à dire tout ce que le logiciel ne sera pas en mesure de faire automatiquement. Certaines recommandations vont donc être ennoncés, qu'il sera préférable de respecter pour le bon fonctionnement de lockatme.

6.2 Utilisation

6.2.1 Installation Qualification

Pour installer le logiciel lockatme, l'utilisateur aura deux options. Le projet sera upload sur PyPI. Il pourra donc utiliser la commande suivante disponible sur toutes les distributions Linux, à condition d'avoir installé le package pip :

```
~$ pip install lockatme
```

Sinon il pourra tout simplement cloner le projet depuis GitHub puis le compiler :

```
~$ git clone https://github.com/lockatme/lockatme-specifications.git
~$ cd lockatme
~$ make install
```

Suite à l'installation de lockatme l'utilisateur pourra lancer le logiciel avec la commande suivante dans son shell :

```
~$ lockatme
```

6.2.2 Explication générale

Une fois l'installation réalisée, comme expliqué dans la partie précédente, la commande `lockatme` permettra d'exécuter le logiciel. Son exécution aura comme effet de verrouiller l'écran. Par défaut l'écran sera recouvert d'un filtre transparent avec un cadenas (voir figure 6.1). A noter que durant cette période de verrouillage, l'écran de l'ordinateur pourra très bien se mettre en veille (écran noir), selon les configurations initiales de l'utilisateur. Pour sortir du verrouillage l'utilisateur aura juste à appuyer sur une touche du clavier. Ensuite la reconnaissance faciale commencera immédiatement (voir figure 6.2). Un point rouge apparaissant et disparaissant indiquera une prise de photo de la part de la webcam. Si une photo prise ainsi correspond à une photo modèle, l'écran se déverrouillera. Au bout d'un moment (configurable dans le fichier de configuration), si le déverrouillage échoue l'utilisateur sera invité à entrer son mot de passe (voir figure 6.3).

FIGURE 6.1 – Ecran verrouillé

FIGURE 6.2 – Déverrouillage écran avec reconnaissance faciale

FIGURE 6.3 – Déverrouillage écran, demande mot de passe

6.2.3 Exemple sous i3

Photos modèles

Le choix des photos modèles devra se faire de manière autonome. À aucun moment le logiciel invitera l'utilisateur à mettre des photos modèles dans le dossier prévu à cet effet. De même, pour le mot de passe. En cas d'absence de photos modèles le logiciel demandera un mot de passe directement. Si le mot de passe n'a pas été défini il suffira d'appuyer sur Entrée. Voici la marche à suivre :

```
~$ cp path/to/image ~/lockatme/images/
```

À noter qu'il est possible de mettre plusieurs photos modèles, dans la mesure ou si l'image de la webcam correspond à l'une des photos le déverrouillage fonctionnera.

Bien évidemment une image de bonne qualité est préférable, le logiciel ne vérifiera pas la qualité de l'image. L'utilisateur devra donc voir à l'usage.

Raccourci clavier

Cet exemple est réalisé à partir d'une distribution ArchLinux avec le Windows manager i3. La marche à suivre est donc susceptible de changer.

```
~$ cd .config/i3/
~/.config/i3/$ vim config
```

Exemple de config :

```
program bindkey
bindsym mod+o exec opera
bindsym mod+g exec chromium
bindsym mod+i exec spotify
bindsym mod+p exec libreoffice
bindsym mod+Shift+f exec firefox
bindsym mod+Key+binding exec lockatme
```

Fermeture clapet

La marche à suivre est la suivante :

```
~$ sudo pacman -S acpid
~$ cd /etc/acpi
/etc/acpi/$ cd events
/etc/acpi/events/$ vim lid
/etc/acpi/events/$ cd ..
/etc/acpi/$ vim lid.sh
```

Il faut tout d'abord intaller acpid. Ensuite à l'intérieur de /etc/acpi/events il faut créer le fichier `lid` (voir ci dessous).

```
event=button/lid LID close
action=/etc/acpi/lid.sh "
```

Ce bout de code va permettre d'exécuter le fichier `lid.sh` (voir ci dessous) contenu dans `/etc/acpi/`.

```
lockatme
```

Ainsi lorsque le clapet se fermera lockatme s'exécutera automatiquement.

6.3 Diagramme

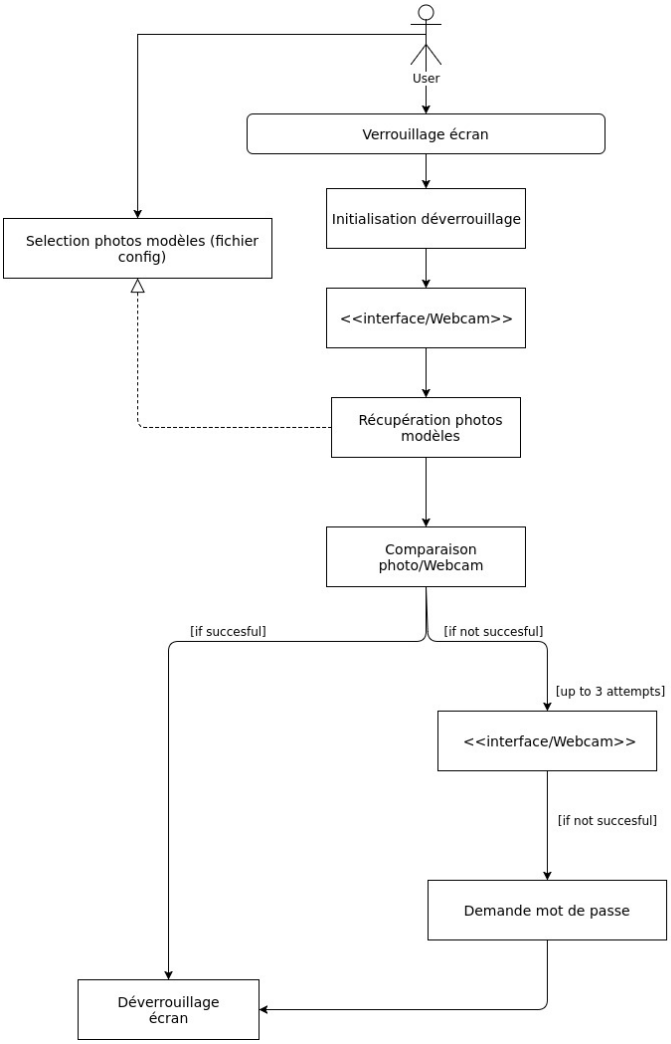

FIGURE 6.4 – Diagramme explication du fonctionnenement général de lockatme

6.4 Bibliothèques utilisées

6.4.1 Contexte

Lorsque nous avons commencé les recherches pour ce projet, nous nous sommes d'abord penché sur l'utilisation de Open_CV, un bibliothèque écrit en C/C++ qui propose des bindings en Python. Open_CV utilise le machine learning pour reconnaître les visages au sein d'une image. La méthode de fonctionnement pour identifier un visage dans une image est dit "en cascade". C'est à dire que le programme va de manières successives et à différentes échelles, vérifier si chaques parcelles de l'image ne contient pas un visage. Sans plus entrer dans les détails, nous avons donc pu écrire quelques lignes de code qui nous permettaient d'identifier les visages au sein d'une image (voir figure 6.5), mais aussi en direct grâce à une webcam. Problème : cette option ne nous permettait pas de simplement d'identifier une personne spécifique dans une image. Chose essentielle pour notre logiciel.

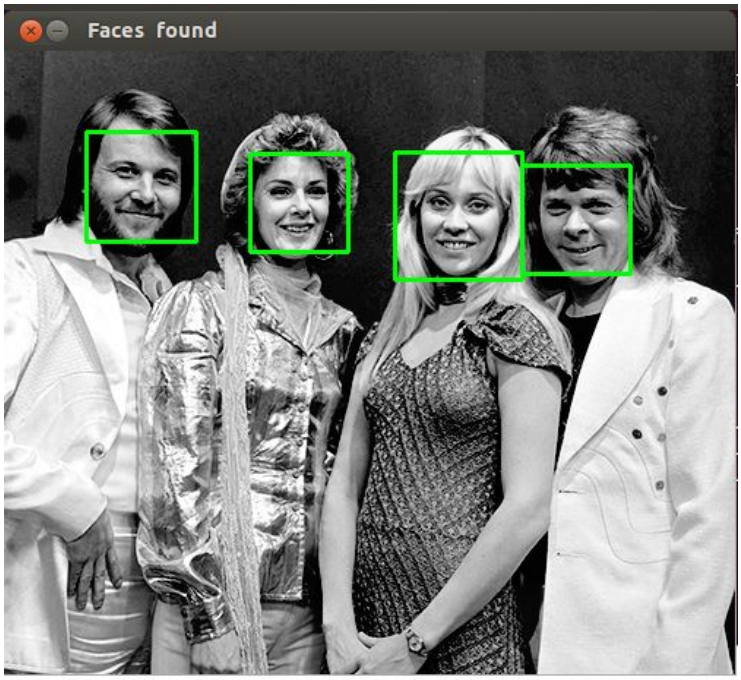

FIGURE 6.5 – Reconnaissance visages avec programme bibliothèque Open_CV

Nous avons par la suite trouvé une autre bibliothèque de plus haut niveau : face_recognition (https ://github.com/ageitgey/face_recognition). Avec celle-ci les possibilités sont diverses, nous avons donc décidé d'utiliser face_recognition pour gérer la reconnaissance faciale de lockatme.

6.4.2 Listing

Lesquelles ? et Pourquoi ?

- Open_CV : accès à la WebCam, nécessaire à la récupération de l'image WebCam
- face_recognition : permet la reconnaissance facial spécifique par Machine learning

6.4.3 Face_recognition

Face_recognition est une bibliothèque très performante qui permet de détecter les visages dans une image et d'identifier les protagonistes en fonction des visages mémorisés (machine learning). La détection se fait grossièrement en 4 étapes :

- Trouver un visage dans l'image
- Analyser les caractéristiques faciales
- Comparer avec les visages connus
- Faire une prédiction sur la personne

La technique de reconnaissance faciale est appelé Histogram of Oriented Gradients (HOG). Tout d'abord les couleurs sont changés pour avoir une image en noir et blanc. Ensuite l'idée est de regarder chaque pixel de l'image et de faire une flèche vers le pixel adjacent le plus sombre. Chaque pixel est ainsi remplacé par une flèche appelé gradient. Ce procédé permet d'avoir une représentation qui est indépendante de la clareté ou de la sombreté de l'image. Mais une flèche par pixel rend l'analyse trop pointue, ainsi l'image du début est séparée en carré de 16x16. Dans chaque carré on trouve une flèche qui représente en moyenne la direction la plus représenté des flèches de chaque pixel en son sein. Ainsi on a une représentation globale et simple du visage (voir figure 6.6 et 6.7)
Pour reconnaître des personnes spécifiques, le logiciel ne va pas chercher à comparer la représentation trouvé par HOG avec toutes les autres représentation trouvé auparavant. Il va réaliser des mesures précises sur les visages trouvés et ainsi émettre une décision par rapport aux mesures enregistrées dans le passé sur d'autre image.

FIGURE 6.6 – Exemple représentation HOG(1)

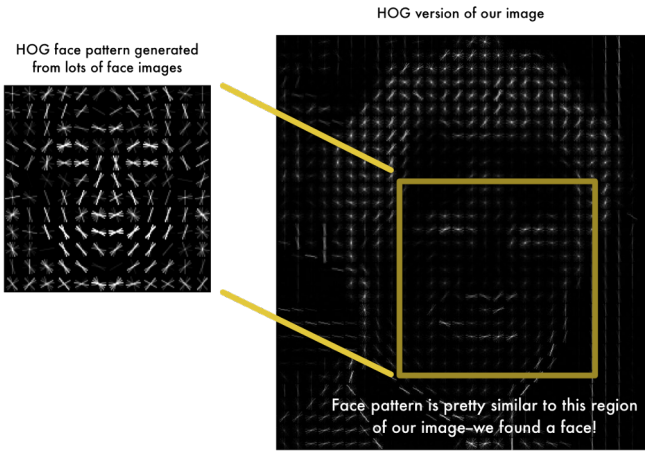

FIGURE 6.7 – Exemple représentation HOG(2)

6.5 Exemple code

Ci dessous le programme qui permet de passer en paramètre une image modèle et une image à reconnaître. Le programme renvoit `true` si la personne sur les deux images est identique et `false` sinon.

```
import face\_recognition as fr


def is_recognized(imageSet, imagePath
    model_image = fr.load_image_file(imageSet
    unknown_image = fr.load_image_file(imagePath

    model_face_encoding = fr.face_encodings(model_image)[0
    unknown_face_encoding = fr.face_encodings(unknown_image)[0

    know_faces = [
        model_face_encoding
    ]

    results = fr.compare_faces(know_faces, unknown_face_encoding)

    return any(results)
```

Le programme va tout simplement prendre l'image modèle et "l'encoder"(HOG) et la placer dans la liste des visages connus. L'autre image est également encodé. L'avant dernière ligne permet de comparer les deux images encodées.

Sur le programme suivant même principe sauf que la reconnaissance se fait en direct avec une Webcam. Le programme permet de dessiner un cadre rouge autour du/des visages et de faire apparaître leur(s) nom(s) si il reconnait un visage. L'encodage se fait ligne 9 et 10.

```
import facerecognition
import cv2
videocapture = cv2.VideoCapture(0)
jujuimage = facerecognition.loadimagefile("juju.jpg")
plimage = facerecognition.loadimagefile("pl.jpg")

jujufaceencoding = facerecognition.faceencodings(jujuimage)[0]
plfaceencoding = facerecognition.faceencodings(plimage)[0]

while True:
    ret, frame = videocapture.read()

    facelocations = facerecognition.facelocations(frame)
    faceencodings = facerecognition.faceencodings(frame, facelocations)

    for (top, right, bottom, left), faceencoding in zip(facelocations,
      faceencodings):
        #See if the face is a match for the known face(s)
        match = facerecognition.comparefaces([jujufaceencoding],
          faceencoding)
        match1 = facerecognition.comparefaces([plfaceencoding],
          faceencoding)
        name = "Unknown"
        if match[0]:
            name = "Juliette"
        if match1[0]:
            name = "PL"
        #Draw a box around the face
        cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

        #Draw a label with a name below the face
        cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0,
          255), cv2.FILLED)
        font = cv2.FONTHERSHEYDUPLEX
        cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255,
          255, 255), 1)

    #Display the resulting image
    cv2.imshow('Video', frame)

    #Hit 'q' on the keyboard to quit!
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
#Release handle to the webcam
videocapture.release()
cv2.destroyAllWindows()
```

Chapitre 7

Implémentation système

7.1 Interface standard

Linux offre une interface standard pour l'authentification à travers la bibliothèque PAM (Pluggable Authentication Module). Des modules peuvent être écrits et utilisés par n'importe quel programme utilisant la bibliothèque. Cela offre le grand avantage d'être très portable et modulaire. Par exemple, de cette manière, MacOS et Android pourraient potentiellement être supporté. En revanche, écrire un module PAM n'est pas des plus aisé. C'est un travail assez bas niveau qui demande une bonne connaissance du ANSI-C. Sans l'existance de bindings Python pour la bibliothèque, cette solution semble être hors de porté pour notre groupe. Cela demanderait d'intégrer notre logique de reconnaissance faciale écrite en Python dans un module PAM, écrit en C. Puis d'utiliser la bibliothèque au sein de notre programme principale pour qu'il fasse utilisation du module créé.

7.2 Screenlocker tier

Une solution alternative serait d'utiliser un screenlocker qui permait de lock et unlock au travers de commandes. Un exemple de programme standard est *XScreenSaver* qui permait de lock avec la commande `xscreensaver-command -l` et d'unlock avec `xscreensaver-command -d`. Cette solution implique d'appeler le programme en question depuis notre propre programme Python. Ceci est moins portable et ne laisse pas autant de liberté, mais est plus simple et plus rapide à implémenter. De plus, cela offre l'avantage d'offrir d'office toutes les fonctionnalités du programme utilisé.

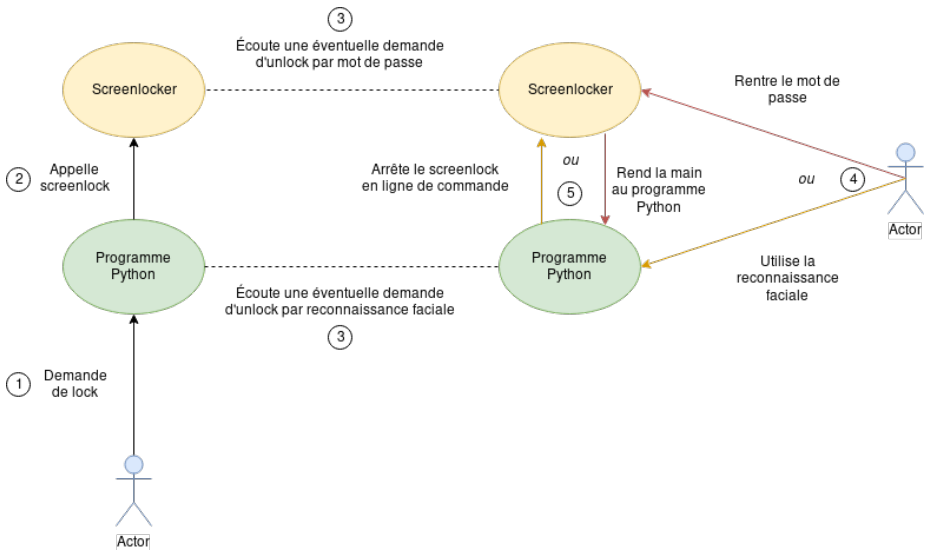

FIGURE 7.1 – diagramme appel programme tier