

Janvier - Juin 2024

RAPPORT DE PROJET

BITUME27

Morgan LUCAS

SNIR - SESSION 2024

LYCÉE LOUIS MODESTE LEROY

TABLE DES MATIÈRES

Présentation	3
Cahier des charges	4
Diagrammes	5-7
Description du système et inventaire du matériels	8
Technologies principales utilisées	9
Énoncé des taches par candidat	10
description des librairies	11
Verification du code de la camera/qr code	12-13
Le thread	14
Présentation JavaSWING	14
Creation de l'application javaswing	15-22
Enregistrement des résultats	23-24
Conclusion et remerciement	24

PRÉSENTATION

Lycée Louis Modeste Leroy
32 rue Pierre Brossolette, 27016 Évreux Cedex
BTS Systèmes Numériques Option A Informatique et
Réseaux Session 2024

Professeurs : ALONSO Stéphane, GOUBIN Thomas,
DENISART Xavier

Le challenge Bitume27 est une association de coureurs qui organise un mini championnat de course à pied dans le département de l'Eure chaque année.

Le but de ce projet est de gérer le classement général du championnat et de faciliter la gestion des courses, via la gestion des inscrit et aussi du déroulé / chronométrage de course. La reconnaissance automatique des coureurs via leur dossards avec un QRCode.

Le système comportera un site internet sur lequel on pourra s'inscrire puis voir les classement. La camera communiquera avec l'application sur ordinateur qui permettra de chronométrer la course, scanner les qr code associé au coureur à l'arriver, et de créer un classement en pdf.

CAHIER DES CHARGES

Le système peut être décomposé en 4 sous-systèmes :

Le sous-système associé à ma partie consiste à avoir un client lourd, un chronométrage et gérer une course.

1) Application lourd qui doit permettre de :

- Visualiser la course et scanner les QR codes (coureurs) à l'arrivée
- Créer/supprimer un classement et le visualiser
- Chronométrer la course à son lancement

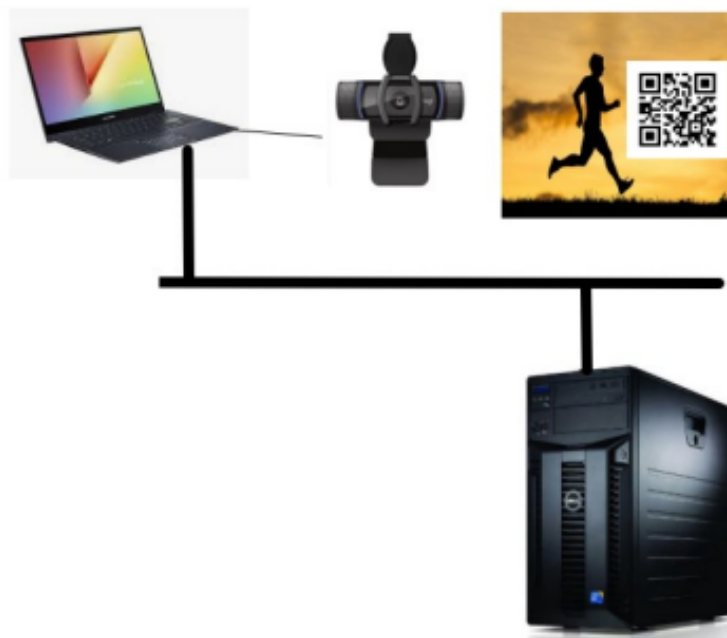


DIAGRAMME DES CAS D'UTILISATION

L'objectif de ce diagramme UML est de représenter les différentes façons dont un utilisateur peut interagir avec un système. Ma partie réside dans la gestion du chronomètre et la réalisation d'un classement.

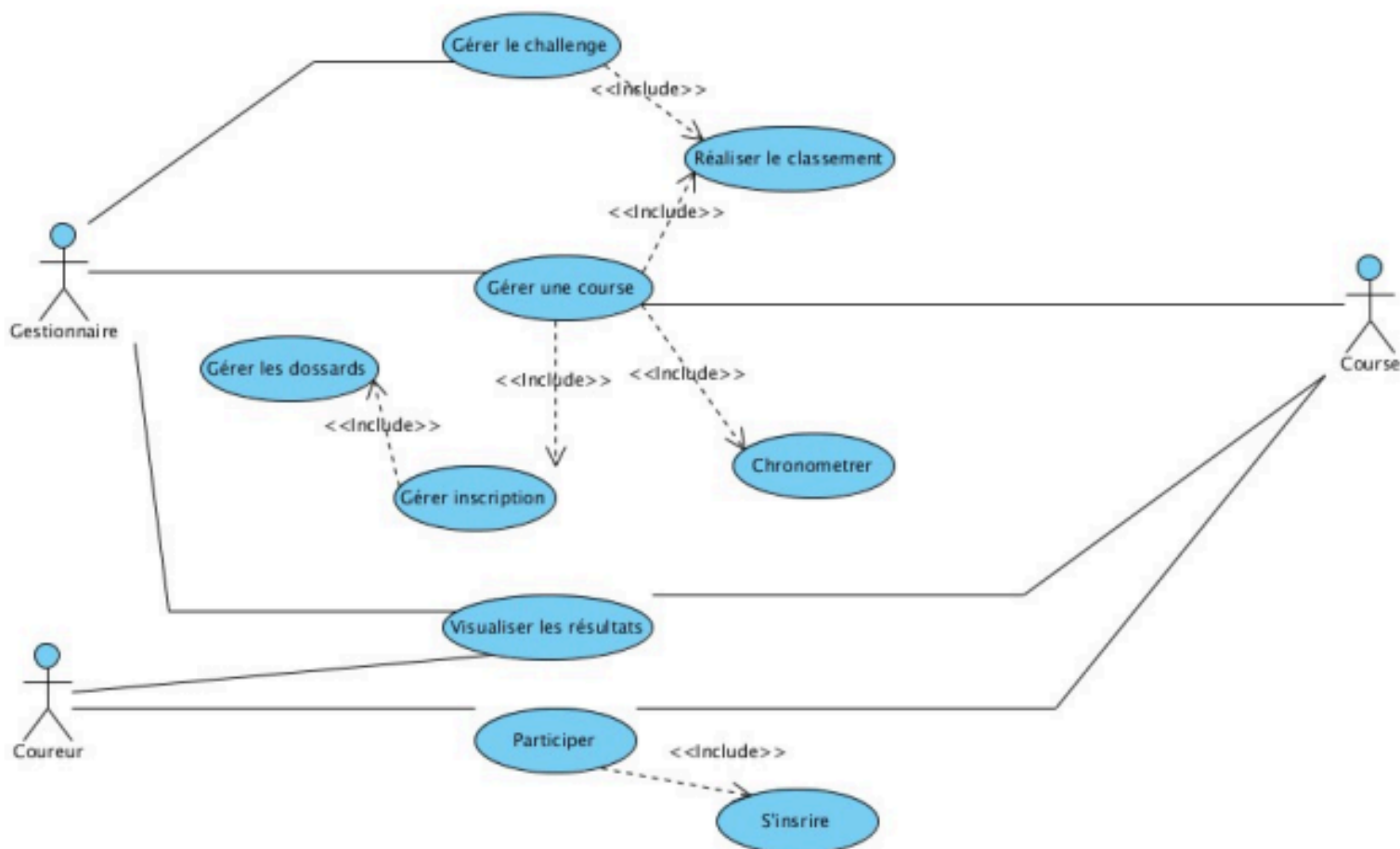


DIAGRAMME DE DÉPLOIEMENT

Le diagramme de déploiement est décomposé en 3 parties pour 3 machines. en effet notre système aura besoin d'un serveur, d'un poste client avec le navigateur web et d'un poste gestion avec le client lourd relier par USB à une caméra, ce poste sera consacré à ma partie avec Clément.

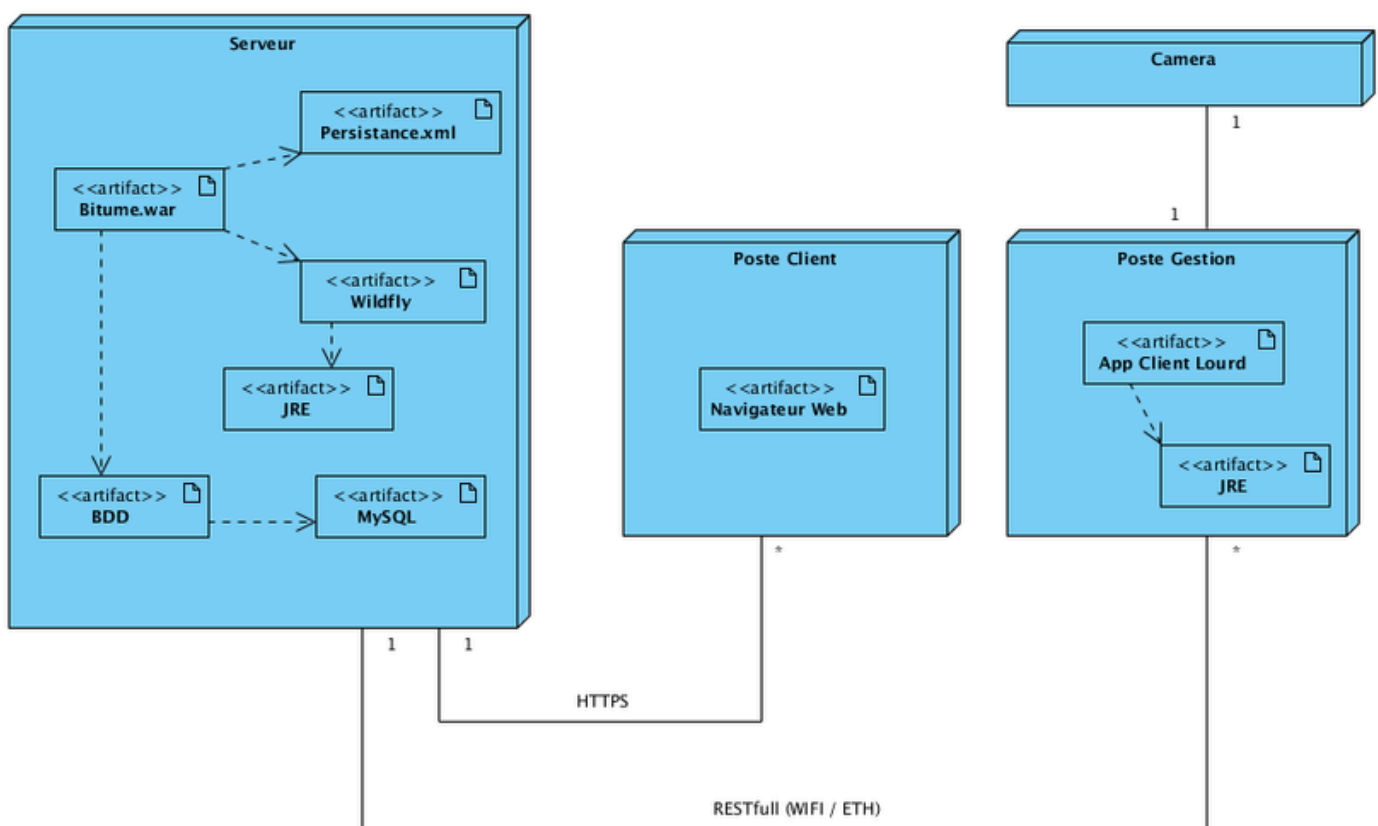
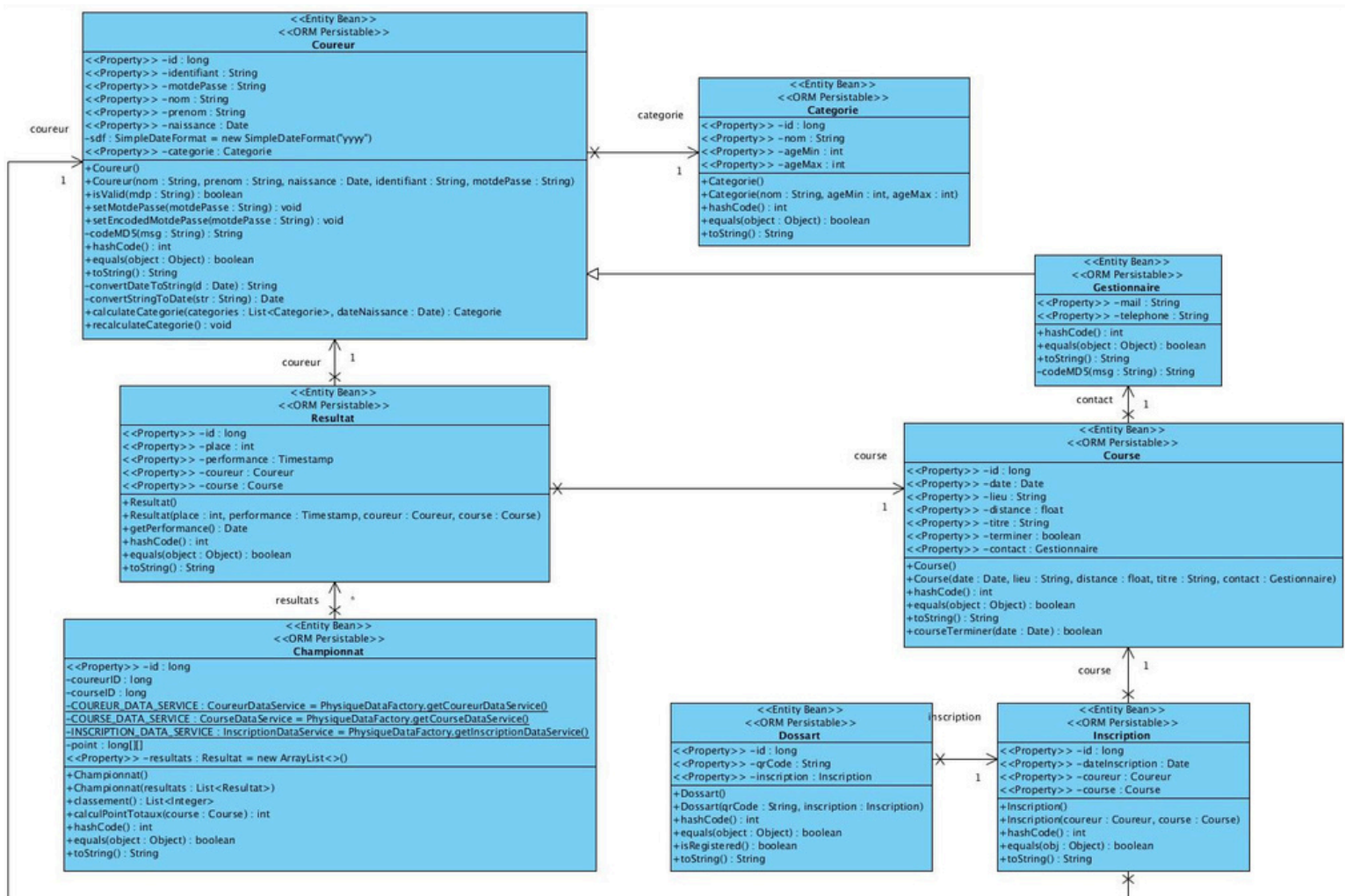


DIAGRAMME DE CLASSE (ENTITÉS)

Un diagramme entité-association est un type d'organigramme illustrant la façon dont des « entités » telles que des personnes, objets ou concepts sont liées les unes aux autres au sein d'un système. Les diagrammes entité-association sont généralement utilisés pour concevoir ou déboguer des bases de données.



DESCRIPTION STRUCTUREL DU SYSTÈME

Principaux constituant:	Caractéristique technique:
<ul style="list-style-type: none"> • PC portable avec Webcam • Chronomètre • Application de gestion de courses • Dossards 	<ul style="list-style-type: none"> • Language JAVA • Reseaux TCP/IP via ethernet ou wifi • Distribution Linux Kernel 5.0 minimum avec serveur AMP et WildFLY minimum

INVENTAIRE DES MATÉRIELS ET OUTILS LOGICIELS

Désignation :	Caractéristique technique:
<ul style="list-style-type: none"> • J2E - WildFLY • WebCAM et QRCode (lib saxos et zxing) • Lib de la section pour la gestion des protocoles RESTFull, JSON-RPC et Persistence. 	<ul style="list-style-type: none"> • VOIR CI-DESSUS

TECHNOLOGIES PRINCIPALES UTILISÉES

1) Application

-Language Java



-Serveur WildFLY



-JavaSWING



-Java Persistence API



-Librairie Sarxos et zxing

-GitHub



-Librairie IText v7.1



ÉNONCÉ DES TACHES PAR CANDIDAT

- 1) Romain DURET
- 2) Antoine FUNGERE
- 3) Clément FAIVRE
- 4) Morgan LUCAS

	Taches à réaliser
Étudiant 1	<ul style="list-style-type: none">• Mise en place de la couche métier (entité et service généraux) ;• Gestion de la base de données (Technologie JPA avec utilisation des Lib de la section) ;
Étudiant 2	<ul style="list-style-type: none">• Réalisation de l'application WEB (Technologie JSF – PrimeFace) :• Frontend client (s'inscrire, consulter le calendrier, les résultats...) ;• Backend (gérer les courses, les coureurs avec dossards, etc..) .
Étudiant 3	<ul style="list-style-type: none">• Mise en place d'un serveur RESTFull pour donner l'accès aux services à distance de manière sécurisé (HTTPS / TOKEN / AUTHENTIFICATION) ;• Mise en place de l'API coté client.
Étudiant 4	<ul style="list-style-type: none">• Réalisation de l'application client lourd de chronométrage ;• Gestion WebCAM ;• Reconnaissance QRCode.

GITHUB

GitHub est une plateforme open source de gestion de versions et de collaboration destinée aux développeurs de logiciels. Git permet de stocker le code source d'un projet et de suivre l'historique complet de toutes les modifications apportées à ce code.

Le but de cette librairie est de permettre d'accéder directement depuis Java aux webcams intégrées ou connectées en USB. Grâce aux bibliothèques fournies, les utilisateurs peuvent lire les images de la caméra et détecter les mouvements.

Il est possible d'y accéder sur GitHub :

<https://github.com/sarxos/webcam-capture>

ZXing (« Zebra Crossing ») est une API pour le traitement des codes QR en Java. Sa bibliothèque comporte plusieurs composants et nous utiliserons le « core » pour la création de codes QR.

C'est une librairie open source qui est accessible sur GitHub aussi :

<https://github.com/sarxos/webcam-capture>

LIBRAIRIE ITEXT

iText est une bibliothèque logicielle qui fournit une interface de programmation partiellement à code source ouvert servant à créer et manipuler des documents PDF. Écrit en langage Java, en .NET (iTextSharp) ainsi qu'en Java compatible avec Android (iTextG)2.

Pour pouvoir lire les QR codes de nos futurs coureurs il va falloir vérifier si le code fourni de la librairie sarxos fonctionne. Nous avons dans un premier temps tout les imports nécessaires à son fonctionnement.

```
import com.github.sarxos.webcam.Webcam;
import com.github.sarxos.webcam.WebcamPanel;
import com.github.sarxos.webcam.WebcamResolution;
```

Puis le code fournie :

```
@Override
public void run() {
    do {
        try {
            Thread.sleep( millis: 100);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        Result result = null;
        BufferedImage image = null;

        if (webcam.isOpen()) {
            if ((image = webcam.getImage()) == null) {
                continue;
            }

            LuminanceSource source = new BufferedImageLuminanceSource(image);
            BinaryBitmap bitmap = new BinaryBitmap(new HybridBinarizer(source));

            try {
                result = new MultiFormatReader().decode( image: bitmap);
            } catch (NotFoundException e) {
                // fall thru, it means there is no QR code in image
            }

            if (result != null) {
                textarea.setText( t: result.getText());
            }
        } while (true);
    }
}
```

La méthode vérifie si il y a une caméra et si elle détecte un QR code, si non alors il ne se passe rien sinon la valeur du QR code et stocké dans la variable "result" et l'affiche dans une zone de texte. Enfin la méthode recommence avec la boucle while.

En ce qui concerne la lecture des QR codes elle se fait avec la librairie zxing et les ligne de code si dessous :

```
Result result = null;
BufferedImage image = null;

if (webcam.isOpen()) {

    if ((image = webcam.getImage()) == null) {
        continue;
    }

    LuminanceSource source = new BufferedImageLuminanceSource(image);
    BinaryBitmap bitmap = new BinaryBitmap(new HybridBinarizer(source));

    try {
        result = new MultiFormatReader().decode(bitmap);
    } catch (NotFoundException e) {
        // fall thru, it means there is no QR code in image
    }
}
```

l'image du QR code va être stocké puis on utilise un "try catch" pour pouvoir savoir si on peut décoder l'image et la stocké dans la variable "result"

Pour pouvoir exécuter ces méthodes parallèlement au code il va falloir utiliser la méthode thread.

Un thread est une unité d'exécution faisant partie d'un programme. Elle fonctionne de façon autonome et parallèlement à d'autres threads. Le principal avantage des threads est de pouvoir répartir différents traitements d'un même programme en plusieurs unités distinctes pour permettre leurs exécutions "simultanées".

Présentation JavaSWING

Swing permet de développer des applications graphiques portables en Java. Cela signifie que ces applications fonctionneront de manière identique quels que soient le système d'exploitation et l'environnement graphique utilisé. Swing a été conçu plus spécifiquement pour créer des applications pour ordinateur de bureau. Il est composé d'un ensemble de classes qui se trouvent dans le package `javax.swing` ainsi que dans ses sous-packages.

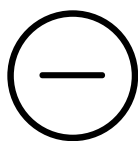
J'utilise cette librairie car j'ai un cours complet dessus et plusieurs exemples de code dans mes travaux précédant.

Point positif et point négatif



richesse des composants proposés

interface graphique produite via Swing sera très bien adaptée pour une application type « client lourd » traditionnel

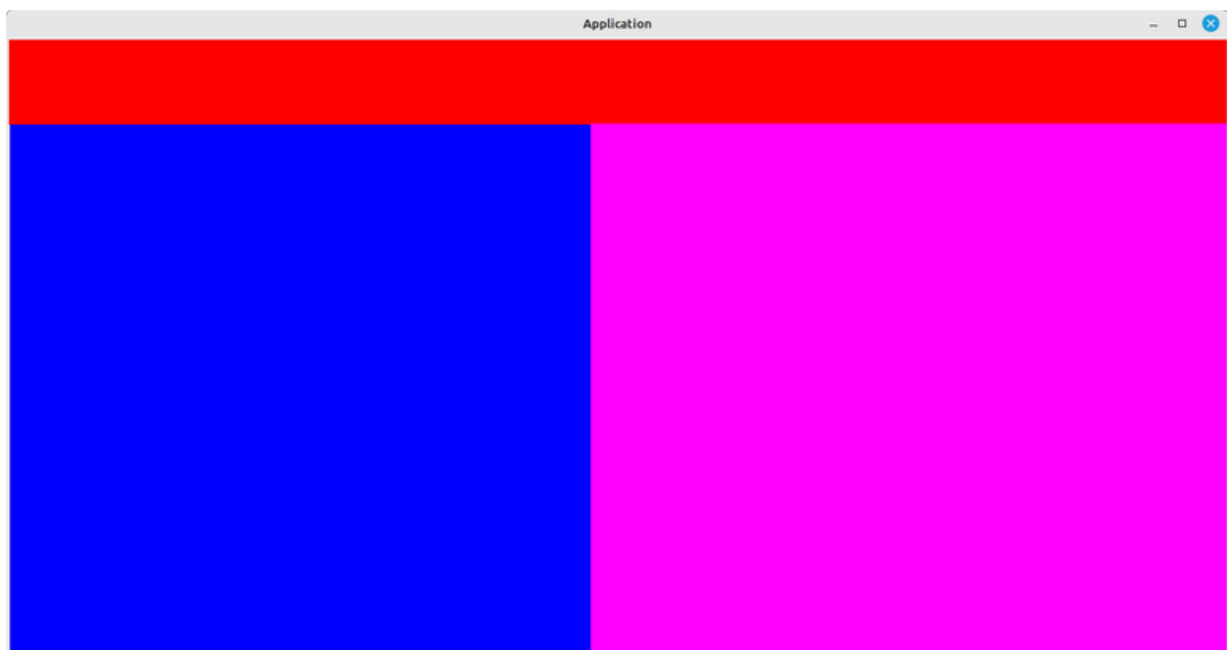


une application codée en Swing consommera beaucoup plus de ressources

Nous avons rapidement vu ce qu'était le JavaSWING je vais maintenant vous montrer à quoi l'application ressemblera en détaillant les différents corps qui la compose.

Le client sera réparti en trois grandes parties.

En rouge il y aura le chronomètre avec les boutons et le compteur, en bleu sera la partie du classement, il y sera affiché le nom et prénom des participants ainsi que leur numéro, le chronomètre et le top. Enfin la partie rose sera l'emplacement où l'utilisateur pourra visionner ce que la caméra voit



L'élaboration d'un chronomètre n'a pas été quelque chose de difficile mais il reste tout de même l'un des points central de ma partie. Ci dessous nous survolerons le code tout en vous expliquant son fonctionnement.

```
public String convertTimeToString() {
    if (millisec > 999) {
        this.millisec = 0;
        sec++;
    }

    if (sec == 60) {
        this.sec = 0;
        min++;
    }

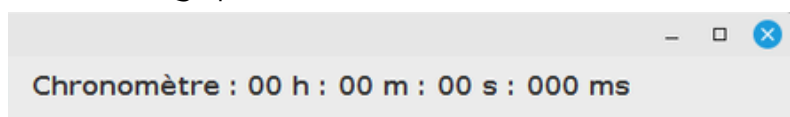
    if (min == 60) {
        this.min = 0;
        hours++;
    }

    return "h : " + hours + ", min : " + min + ", sec : " + sec + ", millisec : " + millisec;
}
```

J'ai simplement utilisé des "if else" en créant des conditions. Nous avons 4 variables à qui on ajoute 1, par exemple. Si **millisec** est plus grande que 999 alors elle retourne à 0 et on ajoute 1 à la variable **sec**, on obtient alors 1 seconde. En faisant pareille pour les autres variables on obtient un chronomètre.

Détaille important, Un ordinateur compte en millièmes de seconde, ce qui veut dire que si nous commençons par les secondes le programme ajouterait 1 à la variable **sec** chaque millièmes de seconde.

À la fin on retourne un string pour avoir sous forme écrite notre chronomètre :



Pour faire fonctionner le chronomètre il a fallut ajouter des JButton. 5 en tout.

Voici exemple de JButton que j'ai fais :

```
JButton button1 = new JButton( string: "Commencer");
button1.addActionListener((e) -> {
    stop = false;
});
button1.setFont(new java.awt.Font( string: "SansSerif", i:1, i1:14));
button1.setBackground(new Color( i:240, i1:145, i2:0));
button1.setForeground( fg: Color.white);
button1.setFocusable( focusable: false);
```

Dans l'exemple j'ai créé un bouton pour lancer le chronomètre que j'ai nommé "**lancement**" auquel j'ajoute un événement quand on interagi avec. Si on clique sur le bouton la variable "**stop = false**" ce qui va arrêter le compteur.

En dessous je me suis amusé à modifier l'apparence du bouton pour qu'il soit plus agréable à regarder, j'ai par exemple changé la couleur avec "**setBackground**".

Ensuite il faut ajouter à notre fenêtre le bouton.

Pour ce faire j'ai donc ajouté dans le JPanel "topPanel" les boutons du chronomètre.

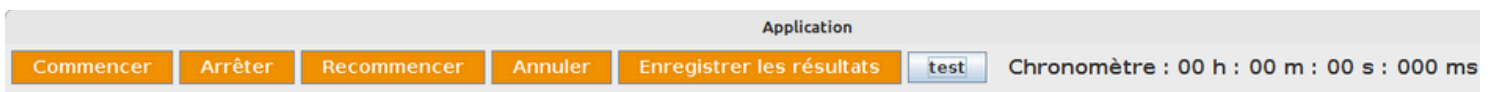
```
//Panel Top
JPanel topPanel = new JPanel();
topPanel.setPreferredSize(new Dimension(1:0, 11:80));
topPanel.setLayout(new FlowLayout(1:FlowLayout.LEFT));
mainContainer.add( comp: topPanel, constraints: BorderLayout.NORTH);

topPanel.add( comp: button1);
topPanel.add( comp: button2);
topPanel.add( comp: button3);
topPanel.add( comp: button4);
topPanel.add( comp: button5);
topPanel.add( comp: button6);
topPanel.add( comp: chrono);
topPanel.add( comp: temps);
```

Le JPanel c'est une balise qui permet de créer un espace qu'on peut paramétrer, comme la taille.

On peut voir ça comme un contenant, et les boutons sont le contenu. Il permet entre autre de créer une zone dans laquelle on peut y ajouter des objets.

Voici le résultat du panel du haut associé au chronomètre.



Pour visualiser une course il nous faut un classement.
JavaSWING est parfait pour ça, avec à la méthode "JTable" qui nous permet la création d'un tableau.

```
//design table
table.setEnabled( enabled: true);
table.setBackground( bg: Color.white);
table.setOpaque( isOpaque: true);
table.setShowVerticalLines( showVerticalLines: false);
table.getTableHeader().setBackground(new Color( i: 240, i1: 145, i2: 0));
table.getTableHeader().setForeground( fg: Color.white);
table.getTableHeader().setFont(new java.awt.Font( string: "SansSerif", i: 1, i1: 14));
table.setRowHeight( rowHeight: 20);

JScrollPane scroll = new JScrollPane( cmpnt: table,
    i: JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,
    i1: JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);

JScrollBar bar = scroll.getVerticalScrollBar();
bar.setPreferredSize(new Dimension( i: 15, i1: 0));
scroll.setPreferredSize(new Dimension( i: 600, i1: 550)); //table size

table.setFillViewportHeight( fillsViewportHeight: true);
```

Une fois le tableau créé, nous pouvons lui ajouter des paramètres telle que sa couleur ou encore sa taille.

Mais pour mettre en forme le tableau et lui ajouter les colonnes que l'on veut il faut créer une "class".

```
public class ModeleDynamique extends AbstractTableModel {
    private final List<Coureur> amis = new ArrayList<Coureur>();

    private final String[] entetes = {"Top", "Temps", "Nom", "Prénom", "QR"};
```

Ici dans notre "class" ModeleDynamique on a un String qui définira le nom de chaque colonne.

Désormais nous avons un tableau qui fait office de classement avec différentes colonnes qui caractérise notre coureur dans notre application.

Top	Temps	Nom	Prénom	QR
classement				

Pour ajouter des valeurs dans notre classement j'ai créé une méthode "scan", cette méthode en appelle une autre "addCoureur".

```
// Scanne et affiche sur le tableau
public void scan() {
    int numDossard = Integer.parseInt(s: qrCodeScan.getText());

    if (checkArriver( numeroCoureur: numDossard)) {
        modele.addCoureur(new Coureur(top++, temps: chronometreTemps.getText(), nom: getNomByDossard( numeroCoureur: numDossard)
            , prenom: getPrenomByDossard( numeroCoureur: numDossard), numero: qrCodeScan.getText()));
        tempsList.add( e: chronometreTemps.getText());
        qrCodeList.add( e: qrCodeScan.getText());
        qrCodeScan = null;
    }
}
```

la méthode "scan" récupère chaque variable pour la méthode "addCoureur" et les affiche grâce à la méthode fireTableRowsInserted.

```
public void addCoureur(Coureur coureur) {
    coureurs.add(e: coureur);
    fireTableRowsInserted(coureurs.size() - 1, coureurs.size() - 1);
}
```

fireTableRowsInserted sert à actualiser le Jtable pour afficher des nouvelles lignes. Elle fonctionne aussi dans le sens inverse avec fireTableRowsDeleted.

```
public void deleteEveryRowInTable(){
    int last = coureurs.size();
    coureurs.removeAll(c: coureurs);
    fireTableRowsDeleted( firstRow: 0, lastRow: last);
}
```

Cette méthode est différente car elle permet de supprimer tout les éléments affichés dans la Jtable.

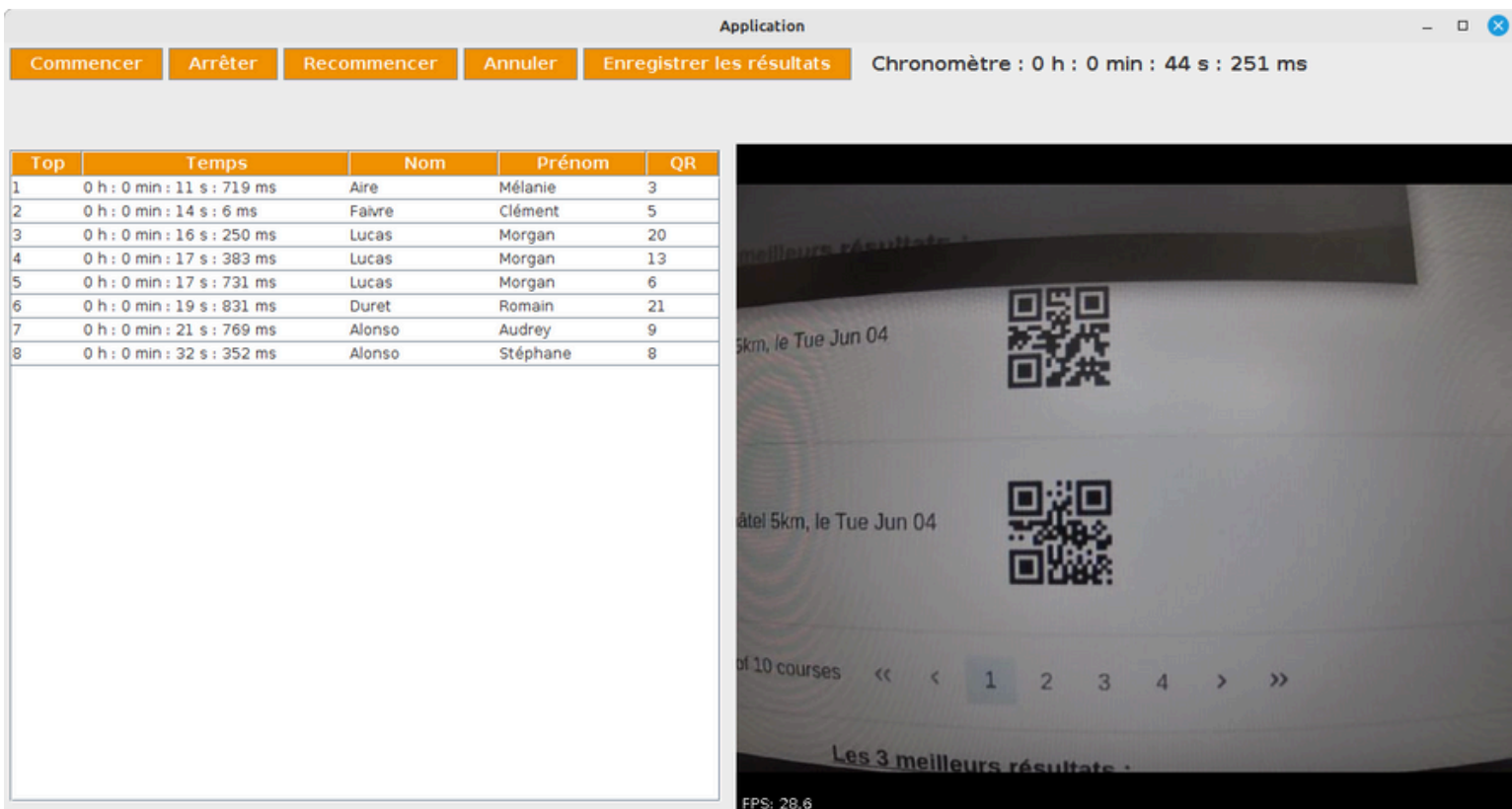
Voici un exemple du classement(table) avec des valeurs inscrites:

Top	Temps	Nom	Prénom	QR
1	0 h : 0 min : 4 s : 732 ms	Duret	Romain	14
2	0 h : 0 min : 8 s : 84 ms	Faivre	Clément	19

Pour rappel, je devais faire une application qui comportait une caméra qui scan des qrCode, un chronomètre et un classement permettant de visualiser l'évolution de la course en temps réel.

Nous avons vu dans un premier temps le fonctionnement de la caméra, puis celui du chronomètre avec les boutons qui permettent son déclenchement et un classement qui récupère les valeurs pour les afficher dans le Jtable.

Nous voila avec une application entièrement fonctionnel, respectant toutes les contrainte du cahier des charges.



The screenshot shows a Java Swing application window titled "Application". It features a toolbar with five buttons: "Commencer", "Arrêter", "Recommencer", "Annuler", and "Enregistrer les résultats". A digital clock displays "Chronomètre : 0 h : 0 min : 44 s : 251 ms".

On the left, a JTable displays the following data:

Top	Temps	Nom	Prénom	QR
1	0 h : 0 min : 11 s : 719 ms	Aire	Mélanie	3
2	0 h : 0 min : 14 s : 6 ms	Faivre	Clément	5
3	0 h : 0 min : 16 s : 250 ms	Lucas	Morgan	20
4	0 h : 0 min : 17 s : 383 ms	Lucas	Morgan	13
5	0 h : 0 min : 17 s : 731 ms	Lucas	Morgan	6
6	0 h : 0 min : 19 s : 831 ms	Duret	Romain	21
7	0 h : 0 min : 21 s : 769 ms	Alonso	Audrey	9
8	0 h : 0 min : 32 s : 352 ms	Alonso	Stéphane	8

On the right, a video feed from a camera shows a QR code scanner interface. It displays two QR codes and text including "5km, le Tue Jun 04", "até 5km, le Tue Jun 04", "ot 10 courses", and "Les 3 meilleurs résultats :". A FPS counter at the bottom left of the video feed shows "FPS: 28.6".

Une fois l'application finie j'ai voulu ajouter une méthode qui permettrait d'enregistrer les résultats sous forme de pdf.

Avec quelques recherches j'ai trouvé une méthode qui le permet grâce à la librairie iText. En voici un exemple :

```
// Création d'un pdf avec les résultats
public void createPdf() {
    System.out.println( x: "Méthode a faire");

    String dest = "/home/jupiter/" + nomPdf + ".pdf";
    PdfWriter writer = null;

    try {
        writer = new PdfWriter( filename: dest);
    } catch (Exception e) {
        e.printStackTrace();
    }

    PdfDocument pdfDocument = new PdfDocument(writer);
    pdfDocument.addNewPage();
    Document document = new Document( pdfDoc: pdfDocument);

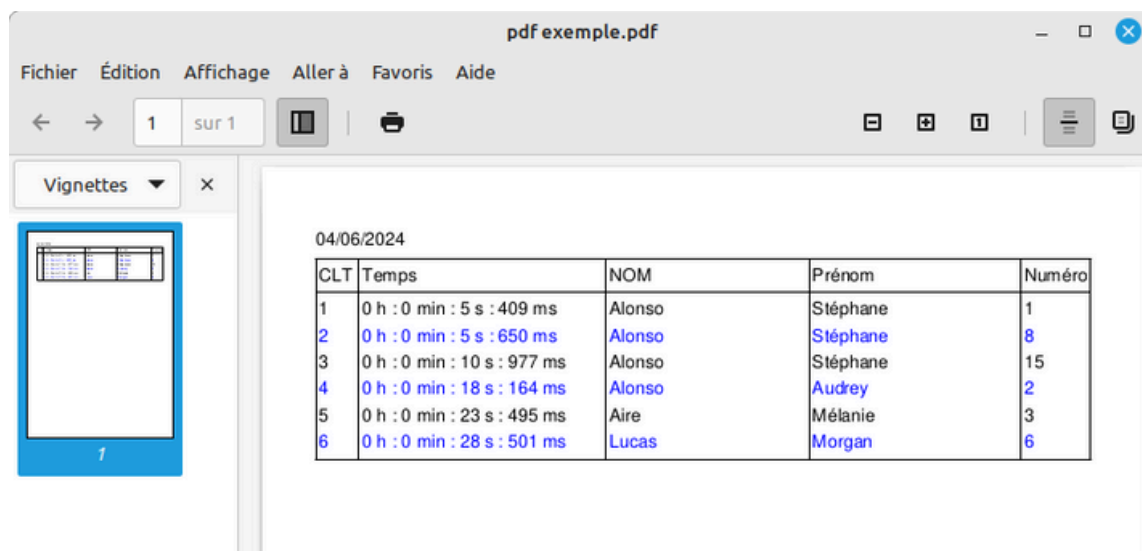
    SimpleDateFormat sdf = new SimpleDateFormat( string: "dd/MM/yyyy");
    document.add(new Paragraph( text: sdf.format(new Date())));

    float[] pointColumnWidths = {30F, 250F, 200F, 200F, 30F};
    Table table = new Table(pointColumnWidths);
```

Pour faire simple nous créons un PDF puis nous lui ajoutons une page pour y insérer du texte. Avec la méthode SimpleDateFormat j'y ajoute la date.

pointColumnWidths sert à définir la taille de chaque colonne. Et nous avons une variable dest qui définit l'emplacement de notre PDF.

Voilà à quoi ressemble actuellement le PDF après avoir enregistré les résultats.



On voit bien la date inscrite et chaque colonne. Les lignes sont séparées par une couleur pour une meilleur lisibilité.

L'apparence peut encore être amélioré cependant le plus important fonctionne.

CONCLUSION ET REMERCIEMENT

La réalisation de ce projet m'a permis d'avoir de meilleur connaissance dans le langage Java et javaSWING. J'ai rencontré plusieurs problèmes qui m'ont permis d'approfondir mes connaissances.

Le travail d'équipe et la mise en relation de toutes les parties du projet auront aussi été très bénéfique.

Je tiens à remercier Mr. Alonso et l'équipe des professeurs pour leurs conseils et leurs aides.