Présentation du projet SafetyNet Alerts

Synthèse client

Classification : Confidentiel

Synthèse client

Rappel de l'objectif

Envoyer des informations aux services d'urgence et espérer ainsi pouvoir donner les moyens aux premiers secours de mieux se préparer et appréhender toutes les situations.

Quelques exemples :
Un incendie se déclare, il faut que SafetyNet Alerts fournisse des informations sur les personnes présentes dans le bâtiment en feu, tels que leurs numéros de téléphone.
Autre exemple : en cas d'alerte ouragan, nous souhaitons que SafetyNet Alerts puisse avertir par SMS tous les habitants de la région. Nous avons donc besoin de recueillir les numéros de téléphone des personnes d'une zone donnée.
Dernier exemple : dans le cas d'une inondation, nous souhaitons fournir aux services d'urgence des informations spécifiques sur les personnes dans la zone. Nous devons donc connaître les victimes potentielles, leurs âges et leurs antécédents médicaux (traitements, allergies, etc.).

- **Besoins** rappel sur les principaux besoins et sur les standards de la société
- **Solutions techniques** – quelques éléments de réponse
- **Logique** – méthode suivie de l'analyse aux tests
- **Modèle** - domaine métier et objets

Classification : Confidentiel

Besoins

- Charger à partir d’un fichier de données l’ensemble des personnes (noms, d'adresses et autres informations) dans notre juridiction locale, ainsi que le fichier de correspondance adresse / casernes de pompiers.
- SafetyNet Alerts doit disposer d’endpoints permettant de conduire à des informations sur leurs statuts. L'ensemble des endpoints que nous aimerions voir au départ est : **health**, **info**, **metrics**, **loggers**.
- SafetyNet Alerts doit disposer d’endpoints permettant aux différentes couches du logiciel d'interagir avec SafetyNet Alerts pour la mise à jour des données Personnes, dossiers médicaux et casernes de pompiers.
- SafetyNet Alerts doit produire différents endpoints en sortie, à partir des éléments chargés. Fichier JSON à partir des URL correspondantes.
- SafetyNet Alerts doit avoir à disposition un ensemble de tests pour valider le bon fonctionnement de l'application. Principe de la pyramide des tests.
- L’application devra logger chaque requête et chaque réponse.
- SafetyNet doit aussi avoir une architecture suivant le motif modèle-vue-contrôleur et répondre aux principes SOLID.

Classification : Confidentiel

Solutions techniques

- **Chargement en mémoire des données principales à partir d'un fichier au démarrage de l'application** Personnes, Station de pompier et Dossiers Médicaux
- **EndPoints statuts**

localhost:5006/actuator/health


```
{
    "status": "UP",
    "components": {
        "diskSpace": {
            "status": "UP",
            "details": {
                "total": 268433354752,
                "free": 196014190592,
                "threshold": 10485760
            }
        },
        "ping": {
            "status": "UP"
        }
    }
}
```


localhost:5006/actuator/info


```
{
    "app": {
        "name": "Spring Sample Application",
        "description": "This is my first spring boot application",
        "version": "1.0.0"
    }
}
```


localhost:5006/actuator/metrics


```
{
    "names": [
        "jvm.threads.states",
        "jvm.memory.used",
        "jvm.gc.memory.promoted",
        "jvm.memory.max",
        "jvm.gc.max.data.size",
        "jvm.memory.committed",
        "system.cpu.count",
        "logback.events",
        "http.server.requests",
        "jvm.buffer.memory.used",
        "tomcat.sessions.created",
        "jvm.threads.daemon",
        "system.cpu.usage",
        "jvm.gc.memory.allocated",
        "tomcat.sessions.expired",
        "jvm.threads.live",
        "jvm.threads.peak",
        "process.uptime",
        "tomcat.sessions.rejected",
        "process.cpu.usage",
        "jvm.classes.loaded",
        "jvm.classes.unloaded",
        "tomcat.sessions.active.current",
        "tomcat.sessions.alive.max",
        "jvm.gc.live.data.size",
        "jvm.gc.pause",
        "jvm.buffer.count",
        "jvm.buffer.total.capacity",
        "tomcat.sessions.active.max",
        "process.start.time"
    ]
}
```


localhost:5006/actuator/loggers


```
{
    "levels": [
        "OFF",
        "ERROR",
        "WARN",
        "INFO",
        "DEBUG",
        "TRACE"
    ],
    "loggers": {
        "ROOT": {
            "configuredLevel": "INFO",
            "effectiveLevel": "INFO"
        },
        "com": {
            "configuredLevel": null,
            "effectiveLevel": "INFO"
        },
        "com.safetynet": {
            "configuredLevel": null,
            "effectiveLevel": "INFO"
        },
        "com.safetynet.alerts": {
            "configuredLevel": null,
            "effectiveLevel": "INFO"
        },
        "com.safetynet.alerts.AlertsApplication": {
            "configuredLevel": null,
            "effectiveLevel": "INFO"
        },
        "com.safetynet.alerts.controller": {
            "configuredLevel": null,
            "effectiveLevel": "INFO"
        },
        "com.safetynet.alerts.controller.EndPointController": {
            "configuredLevel": null,
            "effectiveLevel": "INFO"
        },
```


Classification : Confidentiel

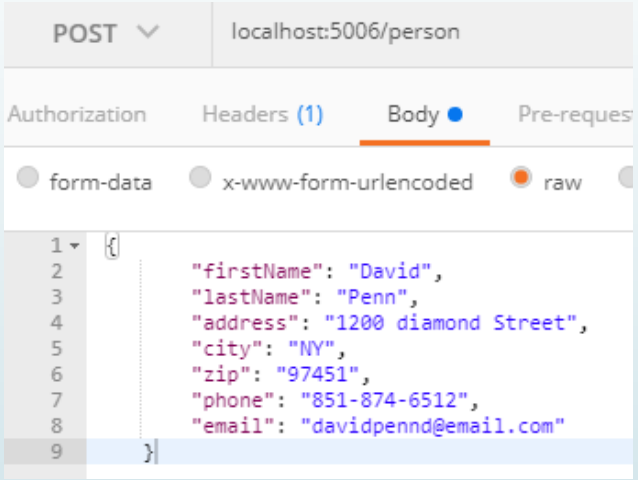
Solutions techniques

Développement des EndPoints pour mise à jour des données durant la session

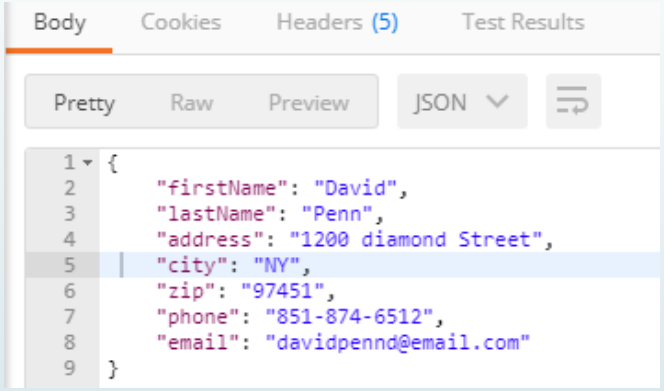
http://localhost:5006/person

- ajouter une nouvelle personne
- mettre à jour une personne existante
- supprimer une personne

Requête POST pour ajout



Réponse Personne créée



LOG requête et réponse

```
2020-04-10 12:22:13.996  INFO 7980 --- [nio-5006-exec-2] c.s.alerts.controller.PersonController   : QUERY_CREATE_PERSON David Penn
2020-04-10 12:22:13.997  INFO 7980 --- [nio-5006-exec-2] com.safetynet.alerts.dao.PersonDaoImpl   : RESPONSE_PERSON_CREATED David Penn
```

http://localhost:5006/fireStation

ajouter une nouvelle caserne-adresse / mettre à jour une caserne-adresse / supprimer une caserne / supprimer une adresse

http://localhost:5006/medicalrecord

ajouter un dossier médical / modifier un dossier médical / supprimer un dossier médical

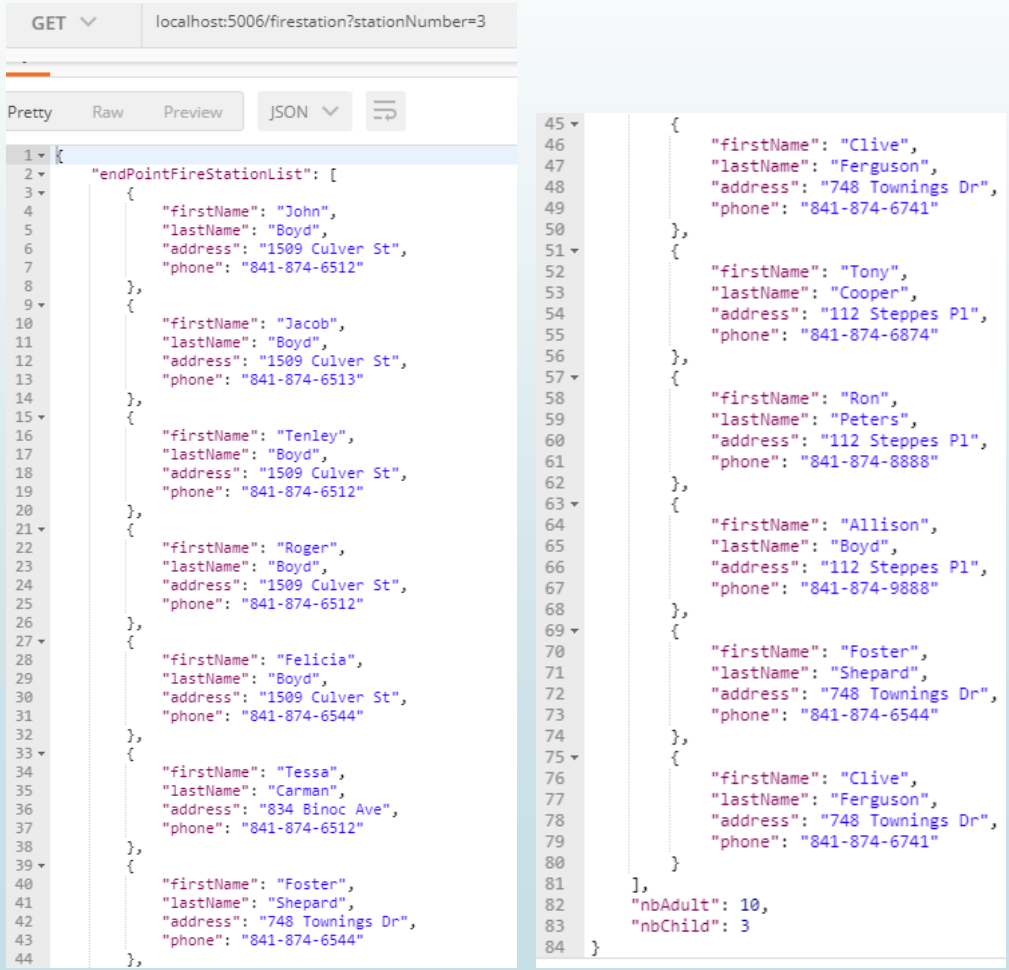
Classification : Confidentiel

Solutions techniques

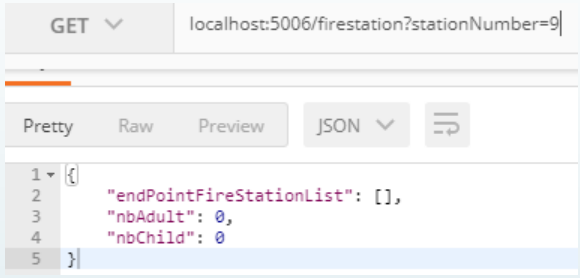
URLs développées pour répondre aux exigences

http://localhost:5006/firestation?stationNumber=<station_number>

Cette url doit retourner une liste des personnes couvertes par la caserne de pompiers correspondante. La liste doit inclure les informations spécifiques suivantes : prénom, nom, adresse, numéro de téléphone. De plus, elle doit fournir un décompte du nombre d'adultes et du nombre d'enfants (tout individu âgé de 18 ans ou moins) dans la zone desservie.



Caserne inexistente



Absence de paramètre

Url inconnue

Classification : Confidentiel

Solutions techniques

URLs développées pour répondre aux exigences

http://localhost:5006/childAlert?address=<address>
Cette url retourne une liste d'enfants (tout individu âgé de 18 ans ou moins) habitant à cette adresse. La liste comprend le prénom et le nom de famille de chaque enfant, son âge et une liste des autres membres du foyer.

http://localhost:5006/phoneAlert?firestation=<firestation_number>
Cette url doit retourne une liste des numéros de téléphone des résidents desservis par la caserne de pompiers. Nous l'utilisons pour envoyer des messages texte d'urgence à des foyers spécifiques.

http://localhost:5006/fire?address=<address>
Cette url doit retourne la liste des habitants vivant à l'adresse donnée ainsi que le numéro de la caserne de pompiers la desservant. La liste inclut le nom, le numéro de téléphone, l'âge et les antécédents médicaux (médicaments, posologie et allergies) de chaque personne.

http://localhost:5006/flood/stations?stations=<a_list_of_station_numbers>
Cette url retourne une liste de tous les foyers desservis par la caserne. Cette liste regroupe les personnes par adresse. Elle inclut le nom, le numéro de téléphone et l'âge des habitants, et fait figurer leurs antécédents médicaux (médicaments, posologie et allergies) à côté de chaque nom.

http://localhost:5006/communityEmail?city=<city>
Cette url retourne les adresses mail de tous les habitants de la ville.

Classification : Confidentiel

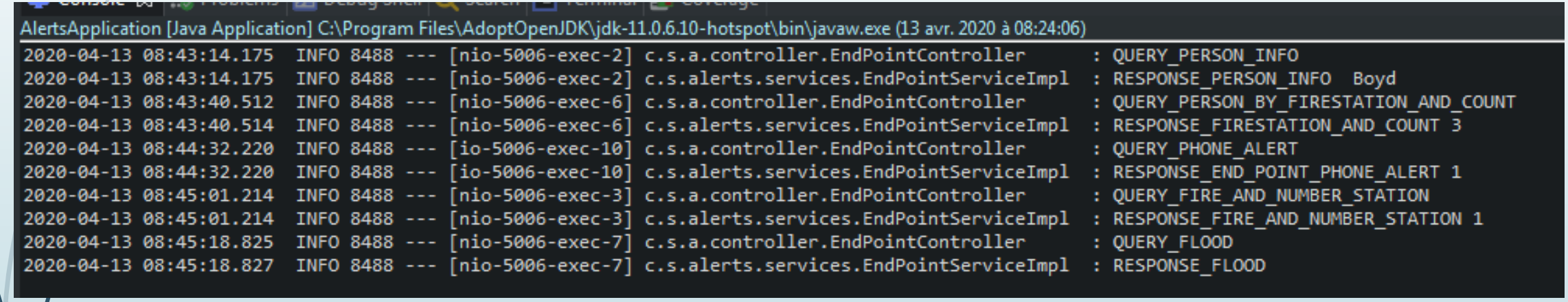
Solutions techniques

URLs développées pour répondre aux exigences

http://localhost:5006/personInfo?firstName=<firstName>&lastName=<lastName>

Cette url retourne le nom, l'adresse, l'âge, l'adresse mail et les antécédents médicaux (médicaments, posologie, allergies) de chaque habitant. Si plusieurs personnes portent le même nom, elles apparaissent toutes.

Affichage des Log à chaque QUERY/RESPONSE



Quelques éléments techniques complémentaires

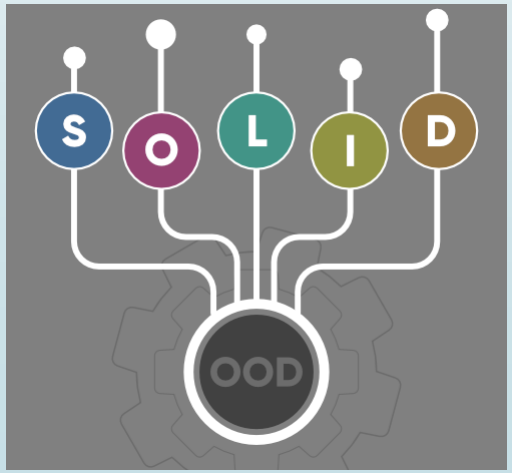
- ✓ Gestion Git avec développement sur branches feature
- ✓ Pucher dans la release GitHub (document du livrable)
- ✓ Un objet réponse par EndPoint – retour JSON
- ✓ Une gestion des anomalies en @RestControllerAdvice pour éviter a gestion des try/catch dans les controlleurs

Classification : Confidentiel

Méthode

- Utilisation du design pattern (Modèle Vue Contrôleur).

La vue correspond au fichier JSON retourné (affichage dans navigateur).
Le contrôleur capte l'action demandée, vérifie la cohérence des données (url et paramètres) et renvoie la réponse.
Le modèle change d'état est constitué par l'ensemble des données ; il reçoit les données et change d'état.



- Application du design pattern SOLID (S et D)

Principe de Responsabilité Unique. Chaque objet est en charge d'une seule responsabilité (raison de changer), laquelle doit être complètement encapsulée dans la classe.
Principe d'inversion des dépendances : Les aspects "bas niveau" sont représentés dans le composant de haut niveau par des interfaces – Les composants "bas niveau" implante les interfaces requises

Classification : Confidentiel

Méthode

Travail sur l'objet personne avec chargement du fichier JSON en mémoire. La méthode de chargement est lancée au démarrage de l'application (@postconstruct).

- Implémentation des interface et classe Dao
- Implémentation de la couche service interface et implémentation
- Développement du contrôleur « personne », méthodes Get, Post, Put et Delete.

Stratégie des tests : configuration de test avec fichier JSON spécifique et un fichier .properties spécifique et activation d'un profil (« test »)

tests unitaires sur la DAO

tests unitaires et intégration sur la couche service (utilisation de Mockito pour les tests unitaires)

tests unitaires et intégration sur le contrôleur (Mockito pour les tests unitaires et MockMvc pour les tests d'intégration)

Interface Service :

expose les méthodes utilisées dans le contrôleur personne

```
public interface PersonService {

    public Person createPerson(Person person) throws DaoCreationException;

    public Person updatePerson(Person person) throws DaoException;

    public Person deletePerson(String firstName,String lastName) throws DaoException;
}
```

Interface Dao ensemble des méthodes d'accès aux données personne

```
public interface PersonDao {

    public List<Person> GetFamilly(String address, String firstName);

    public List<Person> getAllPerson();

    public void SetAllPerson(List<Person> listPerson);

    public Person createPerson(Person person) throws DaoCreationException;

    public Person updatePerson(Person person) throws DaoException;

    public Person getPerson(String firstName, String lastName) throws DaoException;

    public Person deletePerson(String firstName,String lastName) throws DaoException;

    public void deleteAllPerson();

}
```

Classification : Confidentiel

Méthode

- Implémentation de l'objet (caserne) FireStation et développement des classes de tests

Pour l'objet « caserne, deux méthodes de suppression ont été implémentées. Une suppression par adresse qui permet de supprimer une adresse de la liste des stations d'incendie. Cette méthode retourne l'adresse supprimée et sa station de rattachement.
Une méthode par station qui supprime l'ensemble des adresses rattachées à la station. Cette méthode retourne la liste des adresses supprimée pour la station.

- Implémentation de l'objet (dossier médical) MedicalRecord et développement des classes de tests

La spécificité de cette classe porte sur la gestion de la date de naissance. Implémentation d'une classe de « désérialisation JSON » et gestion d'un objet MedicalRecordForReturnFormat pour affichage de la date au format « dd/MM/yyyy ».

- Développement des 7 EndPoints attendus. Chaque réponse du contrôleur correspond à une classe du modèle. Implémentation d'un contrôleur spécifique. L'intégralité du code est développé dans la couche service.

Implémentation des tests unitaires et intégration
Rapport de test surefire et rapport de couverture de test Jacoco
Documents joint dans livrable

```java
public interface EndPointService {

    public EndPointEmail getEmailsByCity(String city);

    public EndPointChildAlert getChildAlert(String address) throws DaoException;

    public EndPointFireStationAndCount getPersonsByFireStation(int stationNumber) throws DaoException;

    public EndPointPhoneAlert getPhoneByStation(int stationNumber);

    public EndPointFireAndNumberStation getFireByAddress(String address) throws DaoException;

    public EndPointFlood getFlood(List<Integer> stations);

    public EndPointPersonInfo getPersonInfo(String firstName, String lastName);
```

Classification : Confidentiel

Modèle du domaine métier

Les objets principaux

Person.java
FireStation.java
MedicalRecord.java

Les objets créés pour répondre aux besoins EndPoints

EndPointEmail.java

EndPointChildAlert.java
EndPointChildAlertDetail.java

EndPointFireStationAndCount.java
EndPointFireStationDetail.java

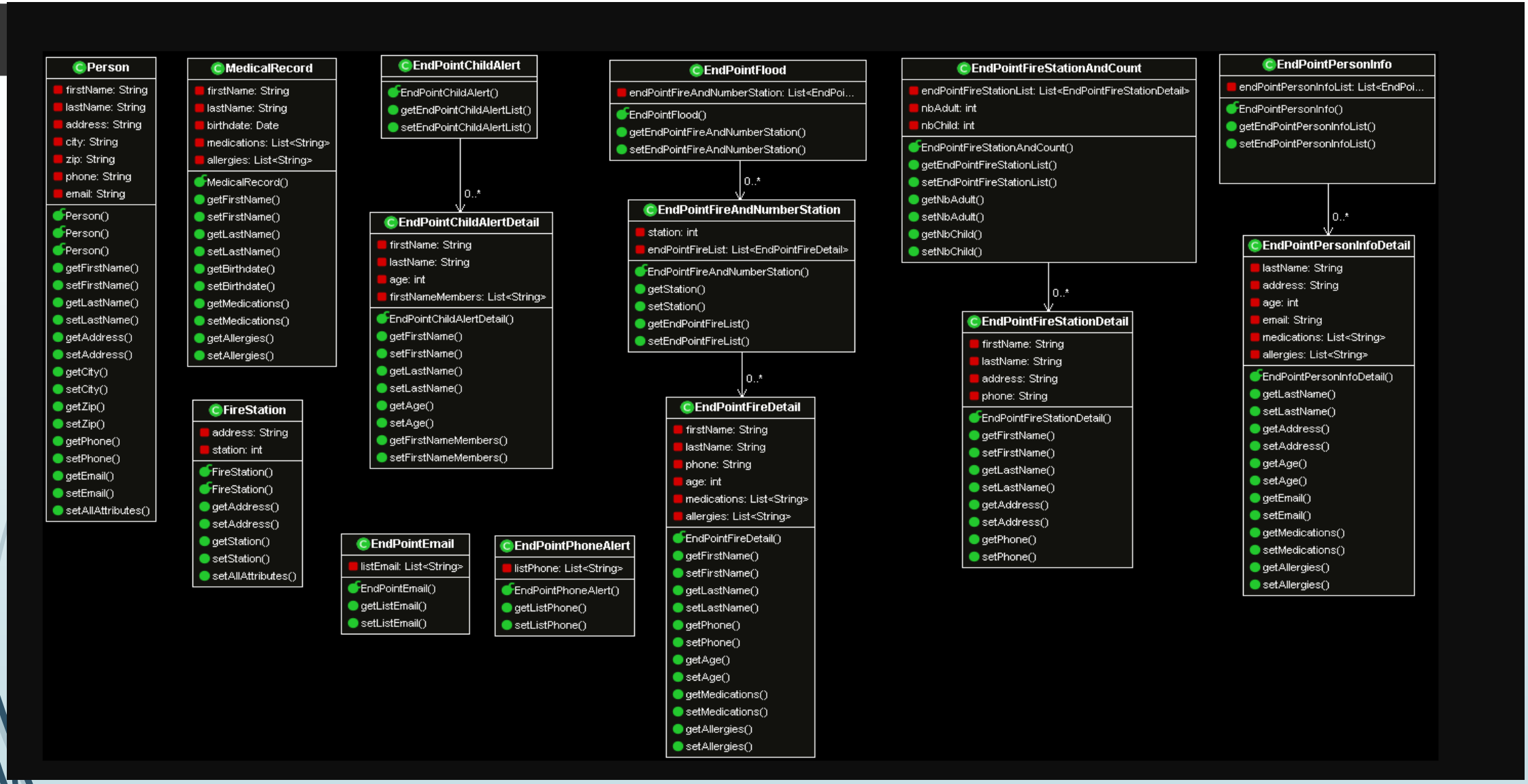
EndPointPhoneAlert.java

EndPointFlood.java

EndPointFireAndNumberStation.java
EndPointFireDetail.java

EndPointPersonInfo.java
EndPointPersonInfoDetail.java

Classification : Confidentiel



Classification : Confidentiel