

GLO-2004 : Génie logiciel orienté objet

Livrable 4

Projet InnoEvent

Rémi Gastaldi – IDUL : REGAS3 – NI : 111 244 692

Léo Hubert – IDUL : LEHUB2 – NI : 111 244 584

Maud Marel – IDUL : MAMAR758 – NI : 111 244 577

Khaled Nasri – IDUL : KHNAS1 – NI : 111 088 348

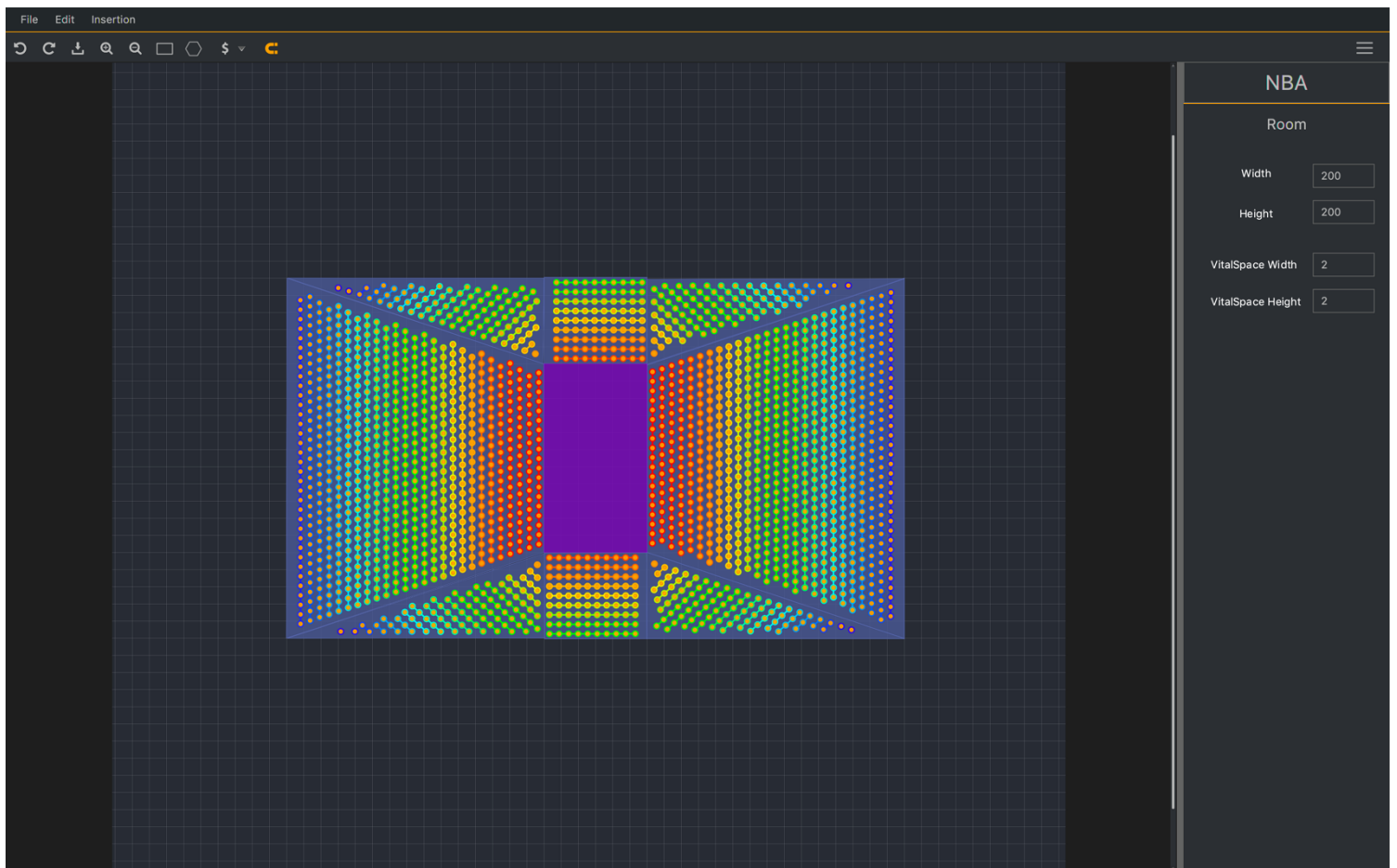
Table des matières

1. Énoncé de vision.....	3
2. Version finale.....	4
3. Modèle du domaine.....	5
4. Modèle des cas d'utilisation.....	8
5. Diagramme de classe de conception.....	9
1.1. UI.....	9
1.1.1. View	10
1.1.2. MainView	10
1.1.3. Pop-up	11
1.1.4. Engine	13
1.1.5. InnoEngine	15
1.2. App	17
1.2.1. Core	18
1.2.2. SaveObject.....	20
1.2.3. InnoSave.....	20
1.2.4. Room	20
1.3. Services	27
1.3.1. Pricing	28
1.3.2. Save	32
6. Conclusion	33
7. Contribution des membres de l'équipe	34

1. Énoncé de vision

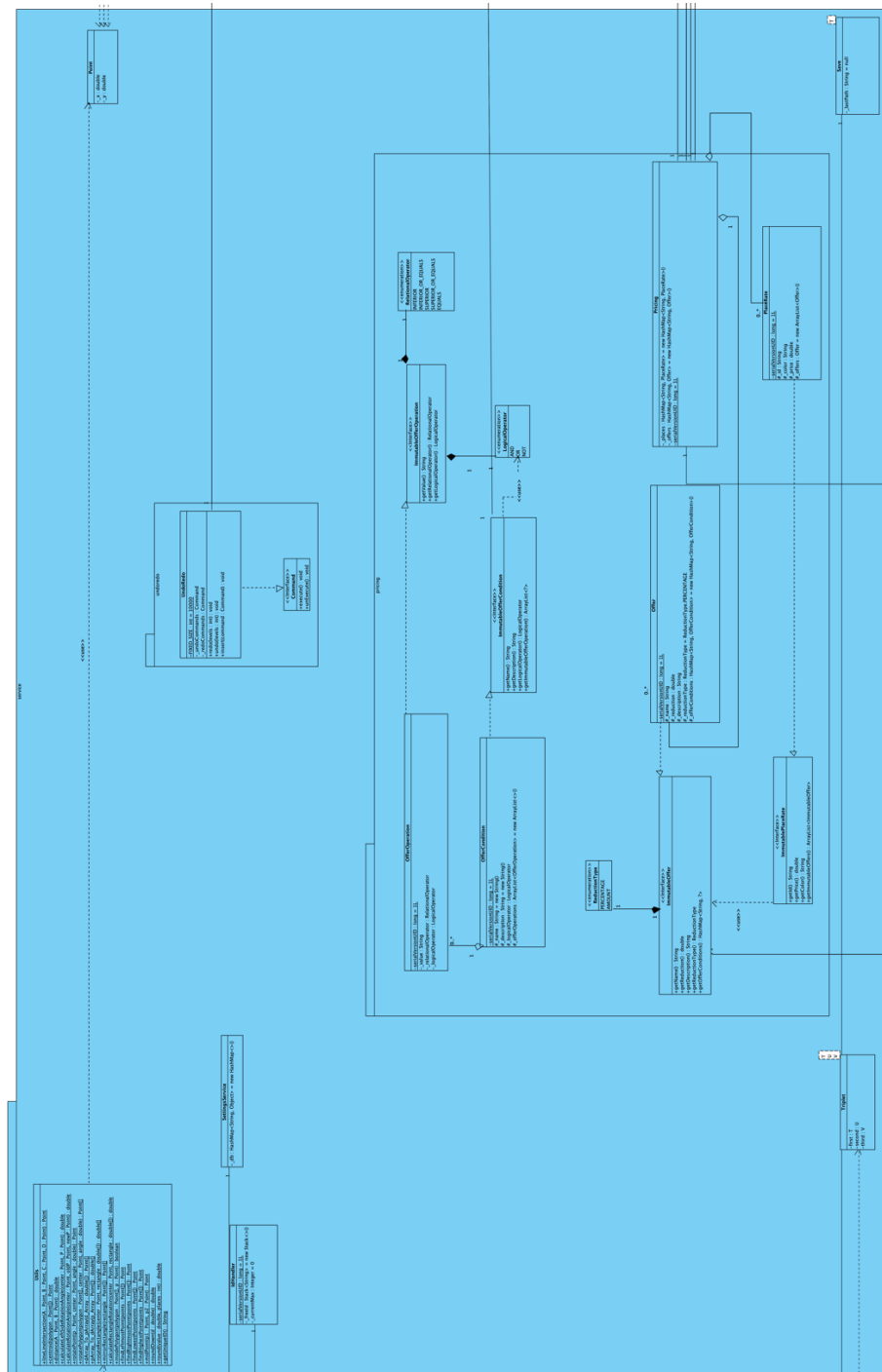
InnoEvent est une application web de création d'évènements permettant de définir le plan de salle pour des concerts, des ballets, des matchs de sport par exemple. L'application permet de configurer le plan de salle de différentes manières grâce à ses différentes fonctionnalités. Ainsi, il est possible d'ajouter des sections réparties autour d'une unique scène. Ces sections peuvent être des sections d'admission générale (debout) ou des sections de siège. Pour ces deux catégories de section, on peut distinguer deux types : les sections régulières correspondant aux sections de forme rectangulaire et les sections irrégulières correspondant aux sections de forme quelconque. Après avoir représenté votre salle avec la scène et ses multiples sections, différentes options pour les prix sont disponibles. Tout d'abord, il est possible d'attribuer les prix manuellement en choisissant d'associer un prix à un ou plusieurs sièges, une ou plusieurs rangées voir même une ou plusieurs sections. Pour cette option, il faut indiquer le prix pour un siège et y associer une couleur si le prix n'a jamais été attribué. Ensuite, la distribution des prix peut être automatisée. Ainsi, il faut juste indiquer le revenu total de la vente, le prix minimum et/ou le prix maximum pour un siège. La répartition des prix est alors réalisée automatiquement en fonction de la position du siège par rapport au centre de la scène. De plus, pour chaque siège(s), rangée(s) ou section(s), une association à une ou plusieurs offres est réalisable. Ces offres peuvent être créées, modifiées, ajoutées, effacées par l'utilisateur. L'utilisateur choisit l'attribution de ces offres en fonction de ses envies et du revenu total souhaité. Cette application permet donc d'avoir un plan de salle conforme à la réalité avec une disposition précise de nos éléments, une répartition des prix modulables selon nos souhaits et une gestion d'offres efficaces.

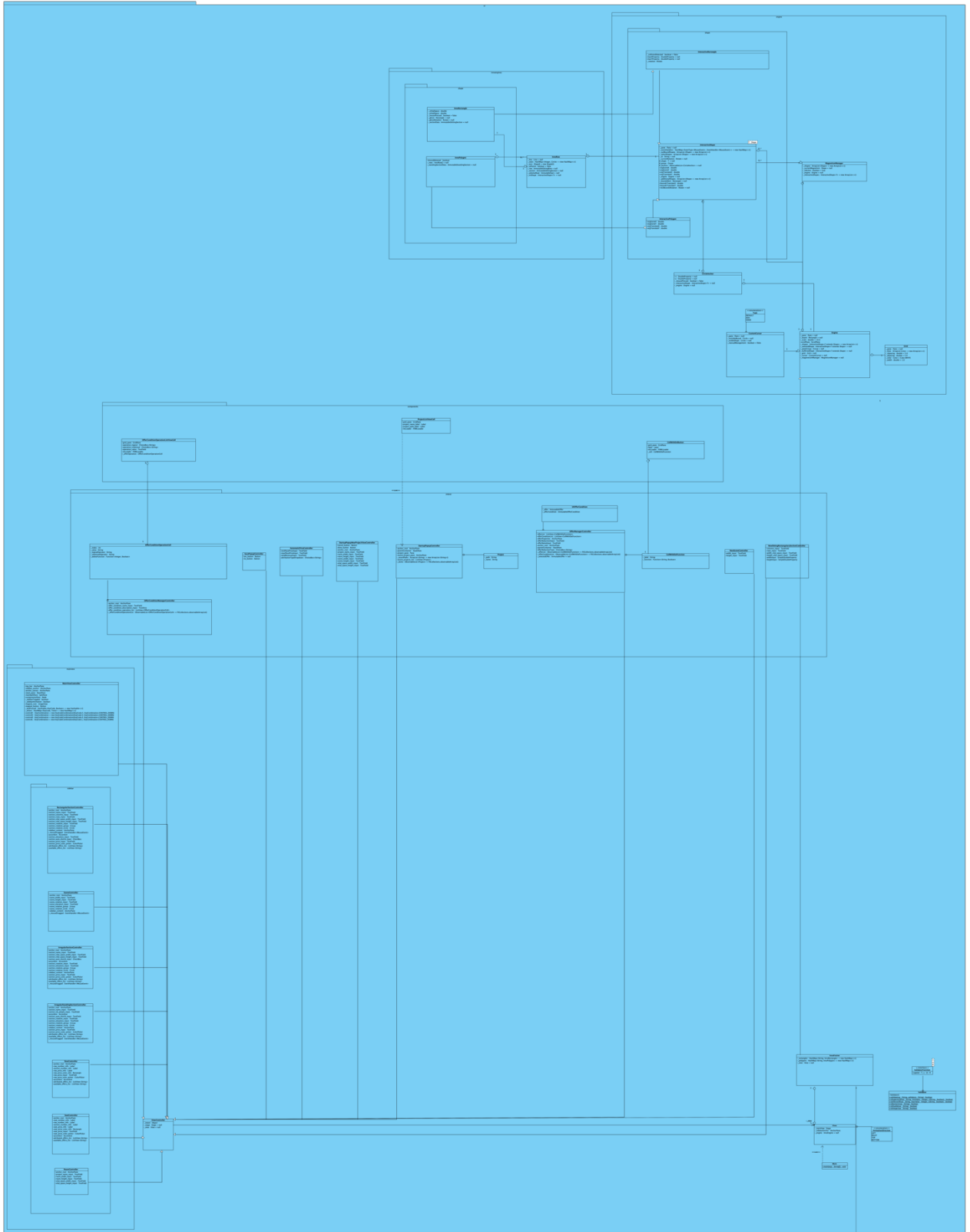
2. Version finale

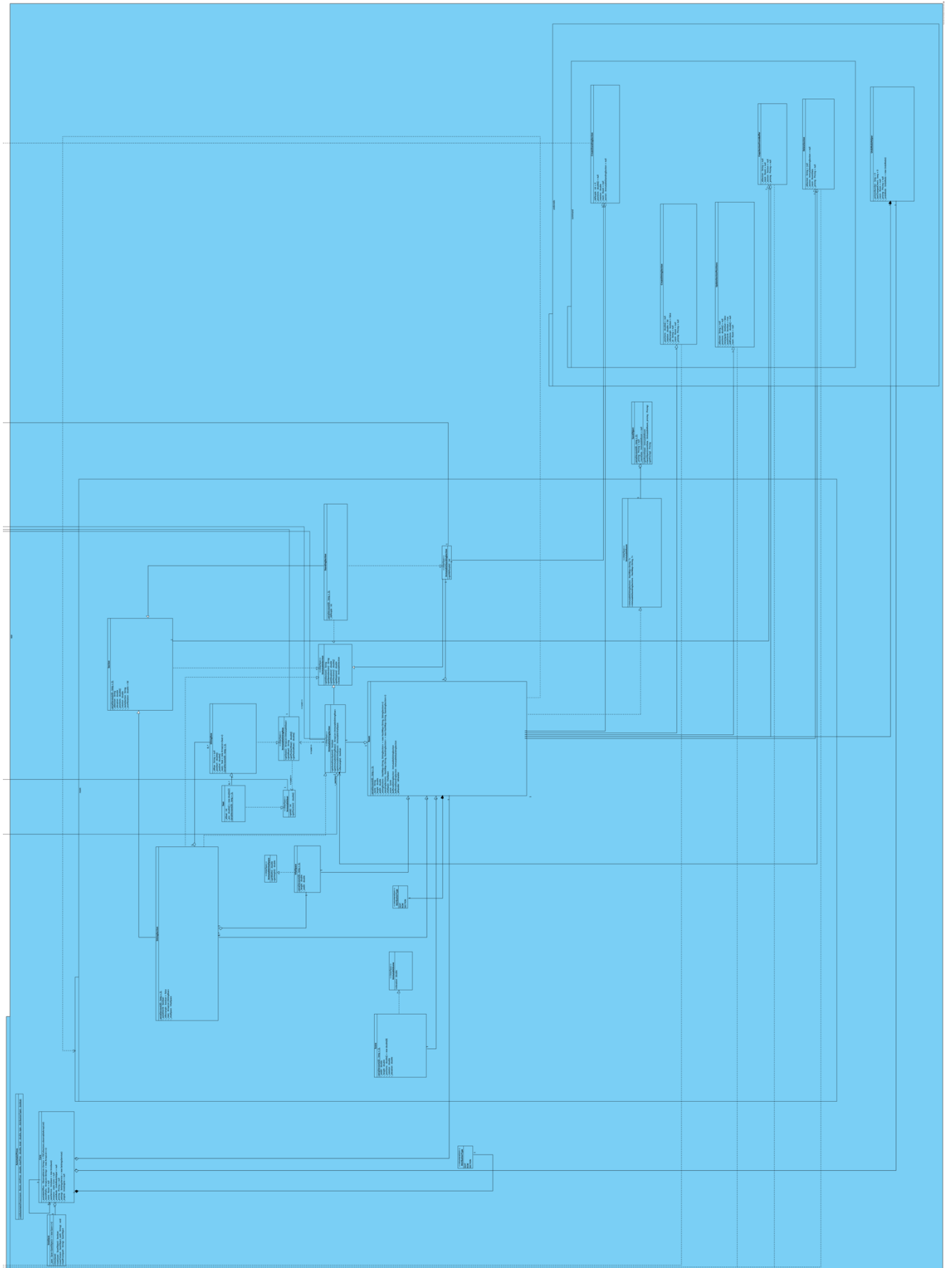


3. Modèle du domaine

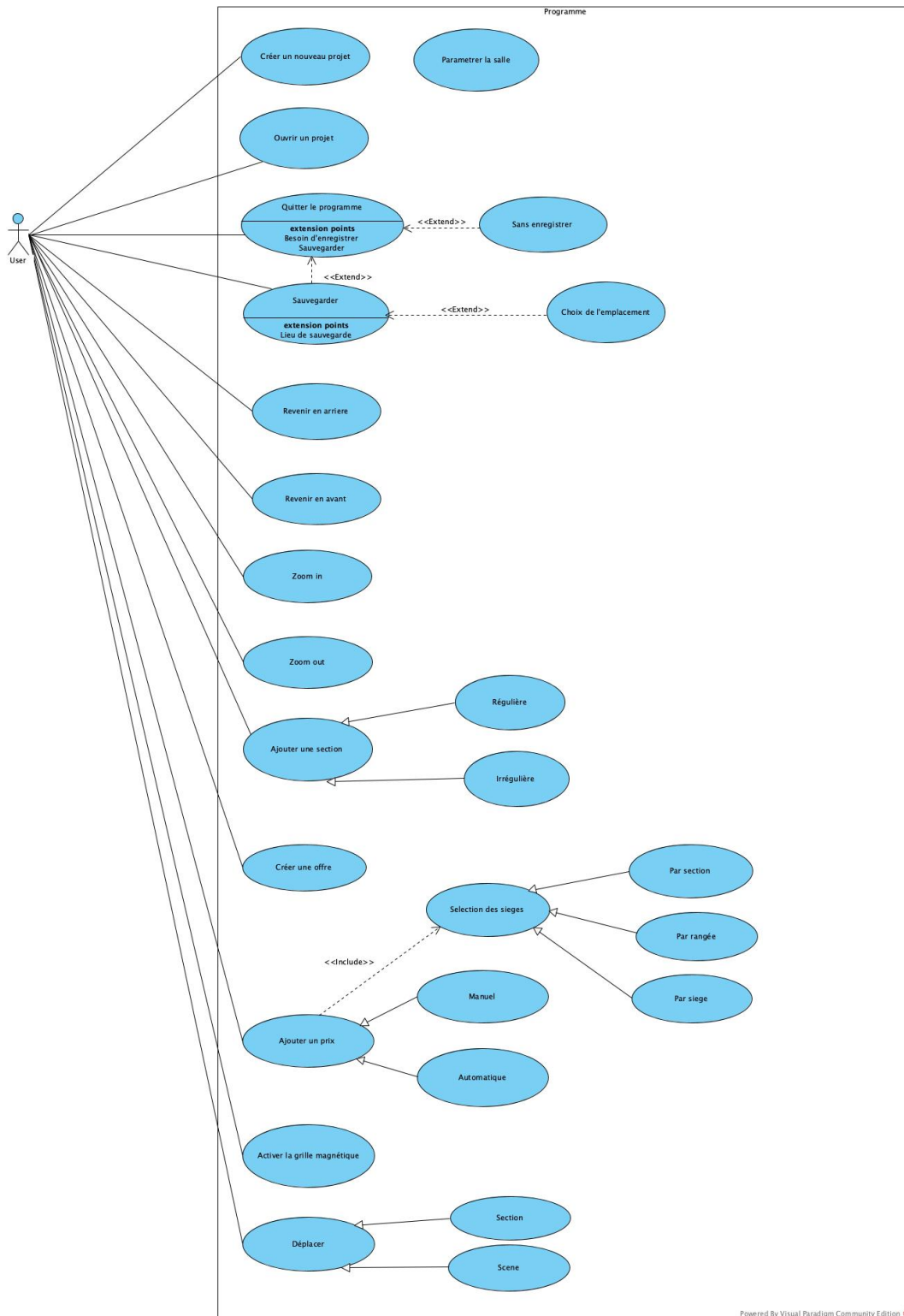
Voir document modele_du_domaine.pdf







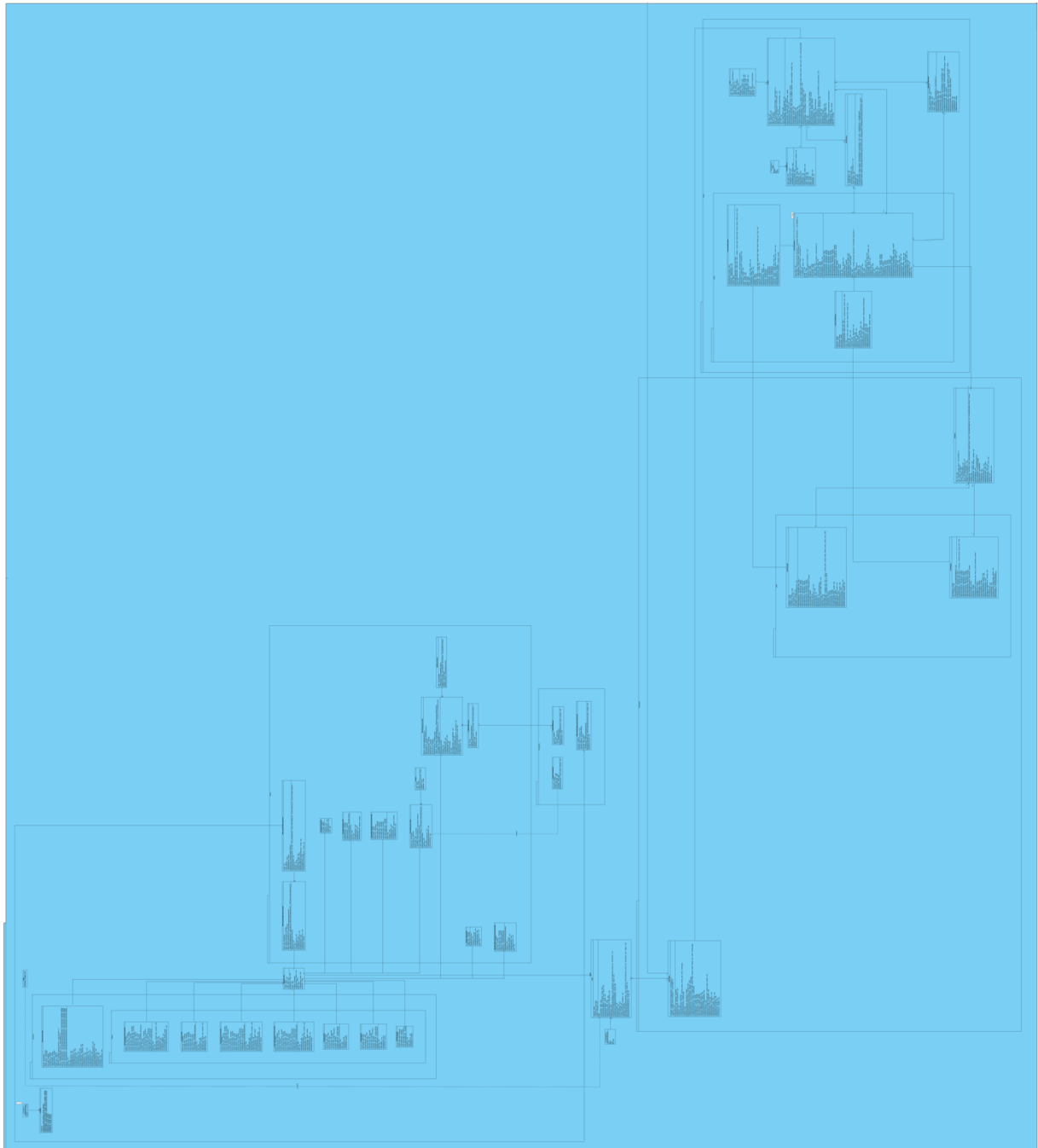
4. Modèle des cas d'utilisation



5. Diagramme de classe de conception

Voir document [diagramme_de_classe_de_conception.pdf](#)

1.1. UI



1.1.1. **View**

Méthodes :

- openView : Permet d'ouvrir une nouvelle fenêtre
- closeView : Permet de fermer une fenêtre
- openViewWithAnimation : Permet d'ouvrir une nouvelle en effectuant une animation
- start : Cette méthode est appelé automatiquement quand JavaFx est prêt et lancé.
- main : Cette méthode est le point d'entrée de notre projet.

1.1.2. **MainView**

1.1.2.1. **MainViewController**

Contrôleur graphique de la fenêtre principale.

Attributs :

- undoButton est un Bouton.
- redoButton est un Bouton.
- saveButton est un Bouton.
- zoomInButton est un Bouton.
- zoomOutButton est un Bouton.
- createIrregularSectionButton est un Bouton.
- createRectangularSectionButton est un Bouton.
- createPriceButton est un Bouton.
- magneticGridButton est un Bouton.
- PaneParent est une Pane.
- scrollPane est une ScrollPane.
- graphicsPane est une Pane.
- sidebarAnchor est une AnchorPane.

Méthodes :

- undoButtonAction : Permet d'annuler l'action précédente.
- redoButtonAction : Permet de refaire la dernière action annulée.
- saveButtonAction : Permet de sauvegarder en passant par le **Core**.
- zoomInButtonAction : Permet de zoomer en avant.
- zoomOutButtonAction : Permet de Zoomer en arrière.
- createIrregularSectionButtonAction : Permet de créer une section de forme irrégulière.
- createRectangularSectionButtonAction : Permet de créer une section de forme rectangulaire.
- magneticButtonAction : Permet d'activer la grille magnétique.

1.1.2.2. **Sidebar**

1.1.2.2.1. **RoomController**

Contrôleur graphique de la classe Room.

Attributs :

- roomHeightInput est un champ de texte.
- roomWidthInput est un champ de texte.

Méthodes :

- saveEventAction change la taille de la room.

1.1.2.2.2. IrregularSectionController

Contrôleur graphique de la création d'une section irrégulière.

Attributs :

- sectionTypeCheckBox est une CheckBox.
- vitalSpaceWidthInput est un champ de texte.
- vitalSpaceHeightInput est un champ de texte.
- elevationInput est un champ de texte.
- automaticDistributionCheckBox est une CheckBox.

Méthodes :

- saveEventAction change les valeurs d'une section irrégulière.

1.1.2.2.3. RectangularSectionController

Contrôleur graphique de la création d'une section rectangulaire.

Attributs :

- vitalSpaceHeightInput est un champ de texte.
- vitalSpaceWidthInput est un champ de texte.
- elevationInput est un champ de texte.
- automaticDistributionCheckBox est une CheckBox.
- nbColumns est un champ de texte.
- nbRows est un champ de texte.

Méthodes :

- saveEventAction change les valeurs d'une section régulière.

1.1.2.2.4. IrregularStandingSectionController

Contrôleur graphique de la création d'une section irrégulière debout.

Attributs :

- sectionTypeCheckBox est une CheckBox.
- maxPeopleInput est un champ de texte.
- elevationInput est un champ de texte.

Méthodes :

- saveEventAction change les valeurs d'une section irrégulière debout.

1.1.3. Pop-up

1.1.3.1. StartupPopupController

Contrôleur graphique de la pop-up de démarrage.

Attributs :

- anchorRoot est une AnchorPane.
- parentContainer est une StackPane.

Méthodes :

- openProjectButtonAction permet d'ouvrir un projet en passant par le **Core**.
- createNewProjectButtonAction permet de créer un projet en passant par le **Core**.

1.1.3.2. **StartupPopupNewProjectViewController**

Contrôleur graphique de la pop-up de création d'un nouveau projet.

Attributs :

- cancelButtonTitle est un Bouton.
- doneButton est un Bouton.
- anchorRoot est une AnchorPane.
- projectNameInput est un champs de texte.
- roomWidthInput est un champs de texte.
- roomHeightInput est un champs de texte.
- sceneWidthInput est un champs de texte.
- sceneHeightInput est un champs de texte.
- vitalSpaceWidthInput est un champs de texte.
- vitalSpaceHeightInput est un champs de texte.

Méthodes:

- doneButtonAction permet de confirmer les informations.
- cancelButtonTitle permet d'annuler la création.

1.1.3.3. **NewPricePopupViewController**

Contrôleur graphique de la création d'un nouveau prix.

Attributs :

- priceInput est un champs de texte.
- colorPickerInput est un champs de texte.

Méthodes :

- doneButtonAction permet de confirmer les informations.
- cancelButtonTitle permet d'annuler la création.

1.1.3.4. **OfferGestionPopupViewController**

Contrôleur graphique de la fenêtre pop-up de gestion d'offres.

Attributs :

- availableOfferListView est un afficheur en liste.
- selectedOfferConditionsListView est un afficheur en liste.
- selectedOfferNameLabel est un label.
- selectedOfferReductionValueInput est un champs de texte.
- selectedOfferNewConditionButton est un bouton.
- selectedOffer affiche l'offre selectionnee.

Méthodes :

- doneButtonAction permet de confirmer les informations.
- cancelButtonTitle permet d'annuler la création.

1.1.3.5. **NewSittingRectangularSection**

Attributs :

- nbRowInput est un champ de texte.
- nbSeatByRowInput est un champ de texte.

Méthodes :

- doneButtonAction permet de confirmer les informations.
- cancelButtonAction permet d'annuler la création.

1.1.3.6. **OfferConditionPopupViewController**

Contrôleur graphique de la pop-up de création d'une nouvelle condition.

Attributs:

- conditionNameInput est un champs de texte.
- conditionValueInput est un champs de texte.
- conditionLogicalOperatorDropDown est un DropDown.
- conditionRelationalOperatorDropDown est un DropDown.

Méthodes:

- doneButtonAction : Permet de confirmer les informations.
- cancelButtonAction : Permet d'annuler la création.

1.1.4. **Engine**

Le package **Engine** est un ensemble de Classes utilitaire permettant de créer interactivement des formes avec l'utilisateur ainsi qu'une grille magnétique au choix, ce package est pensé pour n'avoir aucun lien avec notre projet et pouvant donc être réutilisable dans un autre.

1.1.4.1. **Engine**

La classe **Engine** est la classe principale de son package, elle permet de facilement créer des formes et de gérer toute la logique interne de javaFX en l'encapsulant

Attributs :

- _pane
- _node
- _shapes
- _grid
- _activateMagnetic

Méthodes :

- Engine(**Pane**): Constructeur de l'objet, prends une **Pane** en paramètre qui est le node sur lequel l'engine va interagir.
- setBackgroundColor(**Color**): Permet de changer la couleur en arrière du pane.
- activateGrid(boolean): Permet d'activer ou non la grille visuelle.
- createInteractivePolygon(): Permet d'instancier un **InteractivePolygon** et démarrer l'interaction utilisateur pour qu'il puisse set les points graphiquement.
- createInteractiveRectangle(): Permet d'instancier un **InteractiveRectangle** et démarrer l'interaction utilisateur pour qu'il puisse set les points.
- addInteractiveShape(**InteractiveShape**): Permet d'ajouter une **InteractiveShape** déjà instancié au moteur de l'engine.
- getPane(): Retourne la **Pane** courante d l'engine.
- checkCollision(**InteractiveShape**): Permet de vérifier si une collision a lieu avec un autre **InteractiveShape**, retourne true en cas de collision, false dans le cas contraire.
- activateMagnetic(boolean): Permet d'activer ou non la fonction magnétique de la grille.

1.1.4.2. Grid

Grid est une classe de gestion de la grille visuelle et magnétique, elle permet de choisir l'espacement, la couleur ainsi que l'accrochage magnétique.

Attributs:

- `_pane`
- `_lines`
- `_xSpacing`
- `_ySpacing`
- `_color`
- `_width`
- `_gridLines`

Méthodes:

- `Grid(Pane)`: Construteur, prends en paramètre le Pane avec lequel interagir.
- `zoom(double)`: Permet de zoomer sur la grille pour mettre à l'échelle.
- `setXSpacing(double)`: Permet de régler l'espacement entre les lignes sur l'axe des X
- `setYSpacing(double)`: Permet de régler l'espacement entre les lignes sur l'axe des Y
- `setColor(Color)`: Permet de changer la couleur des lignes de la grille.
- `setLinesWidth(double)`: Permet de changer l'épaisseur des lignes.
- `activate()`: Permet d'activer la grille.
- `disable()`: Permet de désactiver la grille.

1.1.4.3. Shape

Package comprenant toutes les classes de forme interactive du package **Engine**, formes, polygon, rectangle etc.

1.1.4.3.1. InteractiveShape

Attributs:

- `_engine`
- `_pane`
- `_collision`
- `_cursor`

Méthodes:

- `interactiveShape(Engine, Pane)`: Construteur, prends en paramètre l'**Engine** et le Pane avec lequel interagir.
- `changeCursorColor(Color)`: Prends une couleur en paramètre pour changer la couleur du curseur.
- `changeCursorShape(Shape)`: Permet de changer la forme du curseur.
- `start`
- `enableCollision`
- `onMouseEntered(MouseEvent)` : Permet de traiter l'action de l'entrée du curseur dans la fenêtre.
- `onMouseExited(MouseEvent)` : Permet de traiter l'action de la sortie du curseur de la fenêtre.
- `onMouseClicked(MouseEvent)` : Permet de traiter l'action d'un clic de souris.
- `onMouseMoved(MouseEvent)` : Permet de traiter l'action d'un mouvement de souris.

- onMousePressed(**MouseEvent**) : permet de traiter le maintien d'un clic de souris.
- onMouseReleased(**MouseEvent**) : Permet de traiter l'action de libérer un click de souris.
- onMouseDragDetected(**MouseEvent**) : permet de traiter l'action de détection d'un glissement du curseur.
- onMouseDragDropped(**MouseEvent**): Permet de traiter l'action de libérer le glissement de souris.

1.1.4.3.2. InteractivePolygon

Attributs:

- _points : Les points du polygone.
- _lines : Les lignes du polygone.
- _polygon

Méthodes:

- InteractivePolygon(**Engine**, **Pane**): Constructeur de l'objet prenant l'engine ainsi que le **Pane** avec lequel interagir.
- start(): Permet de commencer la création d'un polygone interactif.
- changeCurrentLineColor(**Color**): Permet de changer la couleur de la ligne actuelle.
- addPoint(**MouseEvent**): Permet d'ajouter un point à la position du clic de la souris.
- updateCurrentLine(**MouseEvent**): Mets à jour la position de la ligne courante en fonction de la position de la souris.

1.1.4.3.3. InteractiveRectangle

Attributs:

- _points : Les points du rectangle.
- _lines : Les lignes du rectangle.
- _rectangle

Méthodes:

- interactiveRectangle(**Engine**, **Pane**): Constructeur de l'objet prenant l'engine ainsi que le **Pane** avec lequel interagir.
- start(): Permet de commencer la création d'un rectangle interactif.
- addPoint(**MouseEvent**): Permet d'ajouter un point à la position du clic de la souris
- updateCurrentLine(**MouseEvent**): Mets à jour la position de la ligne courante en fonction de la position de la souris.

1.1.5. InnoEngine

1.1.5.1. InnoEngine

InnoEngine est une classe héritant de la classe Engine pour pouvoir étendre ses fonctionnalités pour pouvoir y rajouter des comportement propre à notre projet.

Attributs:

- _selectedShapeld: id de la forme sélectionnée.
- _polygons
- _rectangles

Méthodes:

- InnoEngine(**View, Pane**): Constructeur de InnoEngine, prends en paramètre les objets nécessaire à la construction de la classe **Engine** dont il hérite.
- createNewProject(String): Permet de lier la création d'un nouveau projet du contrôleur graphique au **Core**.
- createIrregularSection: Permet d'instancier un **InteractivePolygon** et démarrer l'interaction utilisateur pour qu'il puisse set les points graphiquement.
- createRectangularSection(): Permet de démarrer la création interactive d'une section de type rectangulaire.
- setBackgroundColor(**Color**): Permet de changer la couleur en arrière du pane
- activateGrid(boolean): Permet d'activer ou non la grille.
- updateRoomInfos(String, double, double): Permet de mettre à jour les informations de la **Room** provenant du contrôleur graphique au niveau de la couche Domaine ainsi que graphique.
- getSelectedSectionInfos(): Retourne les informations de la section actuellement sélectionnée.
- updateSelectedSectionInfos(): Permet de mettre à jour les informations provenant du contrôleur graphique pour la section actuellement sélectionné au niveau de la couche Domaine ainsi que graphique.
- loadProject(): Permet de lier le chargement d'un projet du contrôleur graphique au **Core**.
- saveProject(String): Permet de lier la sauvegarde du contrôleur graphique au **Core**.
- undo(): Permet d'annuler la dernière action.
- redo(): Permet de refaire la dernière action annulée.
- zoomIn(): Permet de zoomer en avant.
- zoomOut(): Permet de zoomer en arrière.
- magneticGrid(): Permet d'activer ou non la grille magnétique.
- doneAction(): Permet de valider l'action en cours.
- cancelAction(): Permet d'annuler l'action en cours.

1.1.5.2. Shape

1.1.5.2.1. InnoPolygon

Méthodes:

- InnoPolygon
- onMouseClicked(**MouseEvent**): Permet de traiter l'action d'un clic de souris
- onMouseMove(**MouseEvent**): Permet de traiter l'action d'un mouvement de souris
- onMouseExited(**MouseEvent**): Permet de traiter l'action de la sortie de la souris de la fenêtre.
- onMousePressed(**MouseEvent**): Permet de traiter l'action de maintenir un clic de souris.
- onMouseReleased(**MouseEvent**): Permet de traiter l'action de libérer le clic de souris.
- onMouseOnDragDetected(**MouseEvent**): Permet de traiter l'action de glisser la souris.

1.1.5.2.2. InnoRectangle

Méthodes:

- InnoRectangle
- onMouseClicked(**MouseEvent**): Permet de traiter l'action d'un clic de souris
- onMouseMoved(**MouseEvent**): Permet de traiter l'action d'un mouvement de souris

- onMouseExited(**MouseEvent**) : Permet de traiter l'action de la sortie de la souris de la fenetre.
- onMousePressed(**MouseEvent**) : Permet de traiter l'action de maintenir un clic de souris.
- onMouseReleased(**MouseEvent**) : Permet de traiter l'action de libérer le clic de souris.
- onMouseOnDragDetected(**MouseEvent**) : Permet de traiter l'action de glisser la souris.

1.2. App



1.2.1. Core

La classe **Core** est notre contrôleur de Larman, elle permet d'effectuer une liaison entre **la couche UI** et **la couche du Domaine** et la couche des Technical Services, elle est statique et à un attribut qui correspond à l'instance d'elle même ce qui signifie que le Core est un Singleton, il est accessible à n'importe quel endroit dans notre Projet

Cette classe possède 3 attributs :

- *static _instance* correspond à une instance de la classe **Core**.
- *_saveService* correspond à une instance de la classe **Save**.
- *_pricingService* correspond à une instance de la classe **Pricing**.
- *_room* correspond à une instance de la classe **Room**.

Elle contient x méthodes:

- *get* permet de récupérer l'attribut *_instance*.
- *getReductionTypePossibilities* appelle la fonction *getReductionTypePossibilities* de la classe **Pricing**
- *getAttributionTypePossibilities* appelle la fonction *getAttributionTypePossibilities* de la classe **Pricing**
- *getLogicalOperatorTypePossibilities* appelle la fonction *getLogicalOperatorTypePossibilities* de la classe **Pricing**.
- *getRelationalOperatorTypePossibilities* appelle la fonction *getRelationalOperatorTypePossibilities* de la classe **Pricing**.
- *createPlaceRate* appelle la fonction *createPlaceRate* de la classe **Pricing**.
- *createOffer* appelle la fonction *createOffer* de la classe **Pricing**.
- *calculatePrice* calcule le prix par automatiquement par apport aux renseignements donnés.
- *calculatePriceFromDistance* calcule un prix par apport à la distance d'une place à la scene.
- *addOfferCondition* appelle la fonction *addOfferCondition* de la classe **Pricing**.
- *addOfferConditionOperation* appelle la fonction *addOfferConditionOperation* de la classe **Pricing**.
- *deletePlaceRate* appelle la fonction *deletePlaceRate* de la classe **Pricing**.
- *getPlaceRate* appelle la fonction *getPlaceRate* de la classe **Pricing**.
- *setPlaceRatePrice* appelle la fonction *setPlaceRatePrice* de la classe **Pricing**.
- *setPlaceRateColor* appelle la fonction *setPlaceRateColor* de la classe **Pricing**.
- *addPlaceRateOffer* appelle la fonction *addPlaceRateOffer* de la classe **Pricing**.
- *removePlaceRateOffer* appelle la fonction *removePlaceRateOffer* de la classe **Pricing**.
- *deleteOffer* appelle la fonction *deleteOffer* de la classe **Pricing**.
- *getOfferData* appelle la fonction *getOfferData* de la classe **Pricing**.
- *setOfferDescription* appelle la fonction *setOfferDescription* de la classe **Pricing**.
- *setOfferReduction* appelle la fonction *setOfferReduction* de la classe **Pricing**.
- *setOfferReductionType* appelle la fonction *setOfferReductionType* de la classe **Pricing**.
- *removeOfferCondition* appelle la fonction *removeOfferCondition* de la classe **Pricing**.
- *removeOfferConditionOperation* appelle la fonction *removeOfferConditionOperation* de la classe **Pricing**.
- *getOfferConditions* appelle la fonction *getOfferConditions* de la classe **Pricing**.
- *setOfferConditionDescription* appelle la fonction *setOfferConditionDescription* de la classe **Pricing**.

- *setOfferConditionLogicalOperator* appelle la fonction *setOfferConditionLogicalOperator* de la classe **Pricing**.
- *getOfferConditionOperations* appelle la fonction *getOfferConditionOperations* de la classe **Pricing**.
- *setOfferConditionOperationValue* appelle la fonction *setOfferConditionOperationValue* de la classe **Pricing**.
- *setOfferConditionOperationLogicalOperator* appelle la fonction *setOfferConditionOperationLogicalOperator* de la classe **Pricing**.
- *setOfferConditionOperationRelationalOperator* appelle la fonction *setOfferConditionOperationRelationalOperator* de la classe **Pricing**.
- *createRoom* appelle le constructeur de la classe **Room**.
- *getImmutableRoom* renvoie l'attribut *_room* en tant que **ImmutableRoom**.
- *setRoomName* appelle la fonction *setName* de la classe **Room**.
- *setRoomHeight* appelle la fonction *setHeight* de la classe **Room**.
- *setRoomWidth* appelle la fonction *setWidth* de la classe **Room**.
- *setRoomVitalSpaceHeight* appelle la fonction *setHeightVitalSpace* de la classe **Room**.
- *setRoomVitalSpaceWidth* appelle la fonction *setWidthVitalSpace* de la classe **Room**.
- *createScene* appelle la fonction *createScene* de la classe **Room** et renvoie un **ImmutableScene**.
- *deleteScene* appelle la fonction *deleteScene* de la classe **Room**.
- *getSceneData* appelle la fonction *getSceneData* de la classe **Room**.
- *setSceneWidth* appelle la fonction *setSceneWidth* de la classe **Room**.
- *setSceneHeight* appelle la fonction *setSceneHeight* de la classe **Room**.
- *setScenePositions* appelle la fonction *setScenePositions* de la classe **Room**.
- *setSceneRotation* appelle la fonction *setSceneRotation* de la classe **Room**.
- *setSectionName* appelle la fonction *setSectionName* de la classe **Room**.
- *setSectionElevation* appelle la fonction *setSectionElevation* de la classe **Room**.
- *updateSectionPositions* appelle la fonction *updateSectionPositions* de la classe **Room**.
- *deleteSection* appelle la fonction *deleteSection* de la classe **Room**.
- *setSectionRotation* appelle la fonction *setSectionRotation* de la classe **Room**.
- *createStandingSection* appelle la fonction *createStandingSection* de la classe **Room** et renvoie un **ImmutableStandingSection**.
- *setStandingNbPeople* appelle la fonction *setStandingNbPeople* de la classe **Room**.
- *createSittingSection* appelle la fonction *createSittingSection* de la classe **Room** et renvoie un **ImmutableSittingSection**.
- *setSittingSectionVitalSpace* appelle la fonction *setSittingSectionVitalSpace* de la classe **Room**.
- *setSittingSectionAutoRedistribution* appelle la fonction *setSittingSectionAutoRedistribution* de la classe **Room**.
- *createSittingRow* appelle la fonction *createSittingRow* de la classe **Room** et renvoie un **ImmutableSittingRow**.
- *deleteSittingRow* appelle la fonction *deleteSittingRow* de la classe **Room**.
- *clearAllSittingRows* appelle la fonction *clearAllSittingRows* de la classe **Room**.
- *createSeat* appelle la fonction *createSeat* de la classe **Room**.
- *loadProject* appelle la fonction *loadProject* du service **Save**.
- *saveProject* appelle la fonction *loadProject* du service **Save**.

1.2.2. SaveObject

La classe **SaveObject** contient les objets qui vont être sérialisée au moment de la sauvegarde.

Elle contient 3 attributs:

- *_room* contient les informations de la salle sous forme de **RoomData**
- *_prices* contient les informations des prix qui ont été créée sous forme de **PlaceRateData**
- *_offers* contient les informations des offres disponible sous form de **OfferData**

Cette class possède 3 methods:

- *getRoomData* permet de récupérer l'attribut *_room*
- *getPlacesRateData* permet de récupérer l'attribut *_prices*
- *getOffersData* permet de récupérer l'attribut *_offers*

1.2.3. InnoSave

La class **InnoSave** permet de sauvegarder n'importe quel type d'objet dans un fichier.

Elle possède 5 methods:

- *setCurrentPath* va modifier le path du fichier du projet dans le service **Save**.
- *getCurrentPaht* va récupérer le path du fichier du projet dans le service **Save**.
- *loadFrom* va charger un fichier et le transformer en **SaveObject**.
- *save* va sauvegarder le projet dans un fichier.
- *saveTo* permet de sauvegarder le projet dans un nouveau fichier.

1.2.4. Room

1.2.4.1. ImmutableRoom

La classe **ImmutableRoom** est une interface de la classe **Room**. Elle possède un ancêtre en Read-only des méthodes de get de la classe **Room** afin de faire transiter les données vers l'**UI**.

Elle possède 8 méthodes :

- *getName*
- *getHeight*
- *getWidth*
- *getImmutableScene*
- *getSectionByld*
- *getImmutableVitalSpace*
- *getImmutableSittingSections*
- *getImmutableStandingSections*.

1.2.4.2. Room

La classe **Room** permet la gestion de la salle. Elle hérite directement de **ImmutableRoom**.

Elle possède 10 attributs :

- *_name* correspond au nom de la salle.
- *_height* correspond à la longueur de la salle.
- *_width* correspond à la largeur de la salle.
- *_scene* correspond à l'objet **Scene**.
- *_vitalSpace* correspond à l'objet **VitalSpace** soit la vitalSpace assignée à toute la salle et par défaut aux sièges lors de leur création.
- *_sittingSections* correspond à un index de **SittingSections** c'est-à-dire des sections assises.
- *_standingSections* correspond à un index de **StandingSection** c'est-à-dire des sections debouts.
- *_idSection* correspond à une liste d'id correspondant aux index qui ont été assigné puis enlevé aux sections lors de leur manipulation et qui pourrait être utilisé lors du nommage des sections à leur création.
- *_idSectionMax* correspond à l'index maximum utilisé lors du nommage d'une section.

Elle possède 36 méthodes :

- *setName* permet de modifier l'attribut *_name*.
- *getName* permet de renvoyer l'attribut *_name*.
- *setHeight* permet de modifier l'attribut *_height*.
- *getHeight* permet de renvoyer l'attribut *_height*.
- *setWidth* permet de modifier l'attribut *_width*.
- *getWidth* permet de renvoyer l'attribut *_width*.
- *setHeightVitalSpace* permet de modifier l'attribut *_height* présent dans l'objet **VitalSpace**.
- *setWidthVitalSpace* permet de modifier l'attribut *_width* présent dans l'objet **VitalSpace**.
- *getImmutableVitalSpace* permet de renvoyer l'objet **ImmutableVitalSpace**.
- *createScene* permet d'appeler le constructeur de la classe **Scene** tout en retournant l'objet **ImmutableScene**.
- *getScene* permet de renvoyer l'attribut *_scene*.
- *deleteScene* permet de supprimer la scène.
- *setSceneWidth* permet d'appeler la méthode *setWidth* de la classe **Scene**.
- *setSceneHeight* permet d'appeler la méthode *setHeight* de la classe **Scene**.
- *setScenePositions* permet d'appeler la méthode *setPositions* de la classe **Scene**.
- *setSceneRotation* permet d'appeler la méthode *setRotation* de la classe **Scene**.
- *getImmutableScene* permet de renvoyer l'objet **ImmutableScene**.
- *getImmutableSectionById* permet de renvoyer l'objet **ImmutableSection** possédant l'index passé en paramètre.
- *setSectionId* permet d'appeler la méthode *setIdSection* de la classe **Section**.
- *getSectionById* permet de renvoyer l'objet **Section** possédant l'index passé en paramètre.
- *setSectionElevation* permet d'appeler la méthode *setElevation* de la classe **Section**.
- *updateSectionPositions* permet d'appeler la méthode *updatePositions* de la classe **Section**.
- *setSectionRotation* permet d'appeler la méthode *setRotation* de la classe **Section**.
- *deleteSection* permet de supprimer une section par rapport à son id.
- *findFreeId* permet de renvoyer un index correspondant au premier index de type nombre à être utilisable pour le nommage d'une section.

- *createStandingSection* permet de créer l'objet **StandingSection**, l'ajoute alors à l'attribut *_standingSections* et retourne un **ImmutableStandingSection**.
- *setStandingNbPeople* permet d'appeler la méthode *setNbPeople* de la classe **StandingSection**.
- *createSittingSection* permet de créer l'objet **SittingSection**, l'ajoute alors à l'attribut *_sittingSections* et renvoie un **ImmutableSittingSection** correspondant à l'objet créé.
- *setSittingSectionVitalSpace* permet d'appeler la méthode *setVitalSpace* de la classe **SittingSection**.
- *setSittingSectionAutomaticRedistribution* permet d'appeler la méthode *setAutoDistribution* de la classe **SittingSection**.
- *createSittingRow* permet d'appeler la méthode *createRow* de la classe **SittingSection** et renvoie un **ImmutableSittingRow**.
- *deleteSittingRow* permet d'appeler la méthode *deleteRow* de la classe **SittingSection** par rapport à son id.
- *clearAllSittingRows* permet d'appeler la méthode *clearAllRows* de la classe **SittingSection**.
- *createSeat* permet d'appeler la méthode *createSeat* de la classe **SittingRow** et retourne un **ImmutableSeat**.
- *getImmutableSittingSections* retourne l'attribut *_sittingSections*.
- *getImmutableStandingSections* retourne l'attribut *_standingSections*

Le constructeur permet d'initialiser la salle en fonction d'un nom, d'une longueur, d'une largeur et d'un espace vital (longueur, largeur).

1.2.4.3. **ImmutableScene**

La classe **ImmutableScene** est une interface de la classe **Scene**. Elle possède un ancêtre en Read-only des méthodes de get de la classe **Scene** afin de faire transiter les données vers l'UI.

Elle possède 5 méthodes :

- *getHeight*
- *getWidth*
- *getPositions*
- *getRotation*
- *getCenter*

1.2.4.4. **Scene**

La classe **Scene** permet la gestion de la scène positionnée dans la salle.

Elle hérite directement de **ImmutableScene** et de **Serializable**.

Cette classe possède 5 attributs :

- *serialVersionUID*
- *_width* correspond à la largeur de la scène.
- *_height* correspond à la longueur de la scène.
- *_positions* correspond aux 4 points délimitant la scène.
- *_rotation* correspond à la rotation de la scène.

Elle possède 9 méthodes :

- *setWidth* permet de modifier l'attribut *_width*.
- *setHeight* permet de modifier l'attribut *_height*.
- *setPositions* permet de modifier l'attribut *_positions*.

- *setRotation* permet de modifier l'attribut *_rotation*.
- *getWidth* permet de renvoyer l'attribut *_width*.
- *getHeight* permet de renvoyer l'attribut *_height*.
- *getPositions* permet de renvoyer l'attribut *_positions*.
- *getRotation* permet de renvoyer l'attribut *_rotations*.
- *getCenter* permet de renvoyer le centre de la scène après calcul grâce à l'attribut *_positions*.

Son constructeur permet d'initialiser la scène en fonction de sa longueur, de sa largeur et de ses 4 points la délimitant.

1.2.4.5. **ImmutableVitalSpace**

La classe **ImmutableVitalSpace** est une interface de la classe **VitalSpace**. Elle possède un ancêtre en Read-only des méthodes de get de la classe **VitalSpace** afin de faire transiter les données vers l'**UI**

Elle possède 2 méthodes:

- *getHeight*
- *getWidth*.

1.2.4.6. **VitalSpace**

La classe **VitalSpace** permet la gestion de l'espace vital des sièges présents dans les sections elle hérite de **ImmutableVitalSpace** et **Serializable**.

Cette classe possède 3 attributs :

- *serialVersionUID*
- *_height* correspond à la longueur de l'espace vital.
- *_width* correspond à la largeur de l'espace vital.

Elle possède 4 méthodes :

- *setHeight* permet de modifier l'attribut *_height*.
- *setWidth* permet de modifier l'attribut *_width*.
- *getHeight* permet de renvoyer l'attribut *_height*.
- *getWidth* permet de renvoyer l'attribut *_width*.

Son constructeur permet d'initialiser un espace vital en fonction de sa longueur et de sa largeur.

1.2.4.7. **ImmutableSection**

La classe **ImmutableSection** est une interface de la classe **Section**. Elle possède un ancêtre en Read-only des méthodes de get de la classe **Section** afin de faire transiter les données vers l'**UI**

Elle possède 4 méthodes :

- *getIdSection*
- *getElevation*
- *getPositions*
- *getRotation*.

1.2.4.8. Section

La classe **Section** permet la gestion des multiples sections présentes dans la salle. Elle hérite directement de **ImmutableSection**.

Cette classe possède 5 attributs :

- *serialVersionUID*
- *_idSection* correspond à l'identifiant, au nom de la section.
- *_elevation* correspond à la valeur de l'élévation de la section.
- *_positions* correspond au tableau des valeurs des points délimitant la section.
- *_rotation* correspond à la rotation de la section.

Elle possède 8 méthodes :

- *setIdSection* permet de modifier l'attribut *_idSection*.
- *setElevation* permet de modifier l'attribut *_elevation*.
- *updatePosition* permet de modifier l'attribut *_points*.
- *setRotation* permet de modifier l'attribut *_rotation*.
- *getIdSection* permet de renvoyer l'attribut *_idSection*.
- *getElevation* permet de renvoyer l'attribut *_elevation*.
- *getPositions* permet de renvoyer l'attribut *_positions*.
- *getRotation* permet de renvoyer l'attribut *_rotation*.

Son constructeur permet d'initialiser une section en fonction de son nom, son élévation, son index et de ses points le délimitant.

1.2.4.9. ImmutableStandingSection

La classe **ImmutableStandingSection** est une interface de la classe **StandingSection**. Elle possède un ancêtre en Read-only des méthodes de get de la classe

StandingSection afin de faire transiter les données vers l'UI

Elle possède 1 méthode :

- *getNbPeople*.

1.2.4.10. StandingSection

La classe **StandingSection** permet la gestion des sections debouts présentes dans la salle.

Elle hérite directement de **ImmutableStandingSection**, de **Section** et de **ImmutableSection**.

Cette classe possède 2 attributs :

- *serialVersionUID*
- *_nbPeople* correspond au nombre maximum de personne pouvant être présent dans la section.

Elle possède 2 méthodes :

- *setNbPeople* permet de modifier l'attribut *_nbPeople*.
- *getNbPeople* permet de renvoyer l'attribut *_nbPeople*.

Son constructeur permet d'initialiser une section debout en fonction de son identifiant (nom), de ses points la délimitant, de sa rotation et du nombre maximum de personne pouvant être présent dans la section.

1.2.4.11. **ImmutableSittingSection**

La classe **ImmutableSittingSection** est une interface de la classe **SittingSection**. Elle possède un ancêtre en Read-only des méthodes de get de la classe **SittingSection** afin de faire transiter les données vers l'UI

Elle possède 3 méthodes :

- *getAutoDistribution*
- *getImmutableSittingRows*
- *getImmutableVitalSpace*.

1.2.4.12. **SittingSection**

La classe **SittingSection** permet la gestion des sections assises présentes dans la salle.

Elle hérite directement de **ImmutableSittingSection**, de **Section** et de **ImmutableSection**.

Cette classe possède 4 attributs :

- *serialVersionUID*
- *_autoDistrib* permet de savoir si il y a une auto distribution des sièges dans la section.
- *_rows* correspond à une liste d'objet **SittingRow** correspondant aux rangées.
- *_vitalSpace* correspond à l'objet **VitalSpace**.

Elle possède 10 méthodes :

- *setAutoDistribution* permet de modifier l'attribut *_autoDistrib*.
- *setVitalSpace* permet de modifier l'attribut *_vitalSpace*.
- *createRow* permet de créer une rangée et de retourner un **ImmutableSittingRow**.
- *deleteRow* permet de supprimer une rangée en fonction de son identifiant.
- *clearAllRows* permet de supprimer toutes les rangées.
- *createSeat* permet d'appeler la fonction *createSeat* de la classe **SittingRow** et de retourner un **ImmutableSeat**.
- *getAutoDistribution* permet de renvoyer l'attribut *_autoDistrib*.
- *getVitalSpace* permet de renvoyer l'attribut *_vitalSpace*.
- *getImmutableSittingRows* permet de renvoyer l'attribut *_rows* en tant que **ImmutableSittingRow**.
- *getImmutableVitalSpace* permet de renvoyer l'attribut *_vitalSpace* en tant que **ImmutableVitalSpace**.

Son constructeur permet d'initialiser une section assise en fonction de son identifiant (nom), ses points la délimitant, sa rotation et son espace vital (longueur et largeur)s.

1.2.4.13. **ImmutableSittingRow**

La classe **ImmutableSittingRow** est une interface de la classe **SittingRow**. Elle possède un ancêtre en Read-only des méthodes de get de la classe **SittingRow** afin de faire transiter les données vers l'UI

Elle possède 4 méthodes :

- *getSeats*
- *getIdRow*
- *getPosStartRow*
- *getPosEndRow*.

1.2.4.14. SittingRow

La classe **SittingRow** permet de gérer les rangées d'une section assise. Elle hérite directement de **ImmutableSittingRow**.

Cette classe contient 4 attributs:

- *_idRow* correspond à l'index de la rangée.
- *_posStart* correspond aux positions du premier siège de la rangée.
- *_posEnd* correspond aux positions du dernier siège de la rangée.
- *_seats* est une liste de **Seat**.

Elle possède 6 méthodes :

- *createSeat* permet de créer un siège et de retourner un **ImmutableSeat**.
- *getSeats* permet de renvoyer l'attribut *_seats*.
- *getImmutableSeats* permet de renvoyer l'attribut *_seats* en tant que **ImmutableSeat**.
- *getIdRow* permet de renvoyer l'attribut *_idRow*.
- *getPosStartRow* permet de renvoyer l'attribut *_posStart*.
- *getPosEndRow* permet de renvoyer l'attribut *_posEnd*.

Son constructeur permet d'initialiser une rangée de siège en fonction de son index, la position de son premier siège et la position de son dernier siège.

1.2.4.15. ImmutableSeat

La classe **ImmutableSeat** est une interface de la classe **Seat**. Elle possède un ancêtre en Read-only des méthodes de get de la classe **Seat** afin de faire transiter les données vers l'UI

Elle possède 2 méthodes:

- *getId*
- *getPosition*.

1.2.4.16. Seat

La classe **Seat** permet la gestion des sièges présents dans une section assise elle hérite de **ImmutableSeat**.

Cette classe possède 2 attributs :

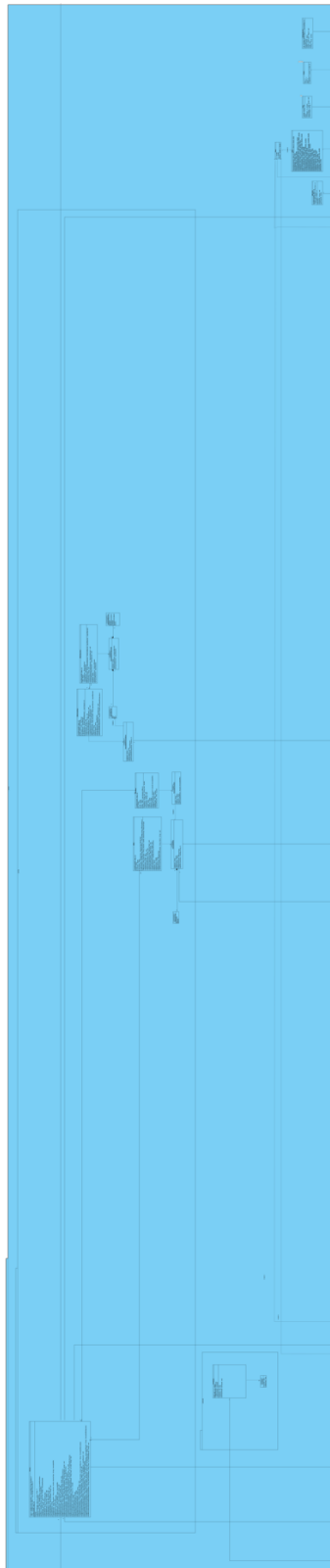
- *_idSeat* correspond à l'index du siège.
- *_pos* correspond à la position du siège.

Elle possède 3 méthodes :

- *setPosition* permet de modifier l'attribut *_pos*.
- *getId* permet de renvoyer l'attribut *_idSeat*.
- *getPosition* permet de renvoyer l'attribut *_pos*.

Son constructeur permet d'initialiser un siège en fonction de son index et de sa position.

1.3. Services



1.3.1. Pricing

1.3.1.1. Pricing

La classe **Pricing** est un service permettant d'attribuer un prix à une place grâce à un identifiant, gérer les offres de celle-ci en créer de nouvelles, gérer les offres existantes, modifier un prix, supprimer le prix d'une place.

Pour nous la classe **Pricing** figure parmi les services car par sa conception elle n'est pas liée à notre projet, elle pourrait être utilisée comme un service par un autre projet de la même manière.

Elle possède 2 attributs:

- *_prices* est un index de **PlaceRate** qui correspond à la liste des prix sur une place qui ont été créés.
- *_offers* est un index d'**Offer** qui correspond à la liste des offres disponibles.
Cette classe contient 4 méthodes:
- *createPlaceRate* permet de créer un prix pour une place avec une couleur, un prix et un identifiant unique afin d'identifier la place, une fois créée, le prix sera contenu dans *_prices*, il sera par la suite possible d'ajouter ou retirer une offre au prix avec les fonctions *addPlaceRateOffer*, *removePlaceRateOffer*.
- *deletePlaceRate* permet de supprimer le prix d'une place grâce à son identifiant.
- *getPlaceRate* permet de récupérer une classe abstraite dont **PlaceRate** hérite qui contient les attributs et les getters nécessaires, celle-ci se nomme **PlaceRateData**.
- *setPlaceRatePrice* permet de modifier le prix d'une place.
- *setPlaceRateColor* permet de modifier la couleur d'une place.
- *addPlaceRateOffer* permet d'ajouter une offre à une place.
- *removePlaceRateOffer* permet de retirer une offre à une place.
- *createOffer* permet de créer une offre ce qui va par la suite l'ajouter dans la liste des offres disponibles à l'ajout d'une place. (*_offers*).
- *deleteOffer* permet de supprimer une offre de la liste des offres disponibles (*_offers*).
- *getOfferData* permet de récupérer une classe abstraite dont **Offer** hérite qui contient les attributs et les getters nécessaires, celle-ci se nomme **OfferData**.
- *setOfferDescription* permet de changer la description d'une offre. (**Offer**)
- *setOfferReduction* permet de changer la valeur de la réduction d'une offre. (**Offer**)
- *setOfferReductionType* permet de changer le type d'une offre, si celle-ci s'appliquera avec un pourcentage ou avec une valeur fixe. (**Offer**)
- *createOfferCondition* permet de créer une condition à une offre.
- *removeOfferCondition* permet de supprimer une condition à une offre.
- *createOfferConditionOperation* permet de créer une opération à une condition qui elle-même est assignée à une offre.
- *removeOfferConditionOperation* permet de supprimer une opération à une condition qui est elle-même associée à une offre.
- *getOfferConditions* permet de récupérer une liste de conditions associées à une offre. (**OfferConditionData**)
- *setOfferConditionDescription* permet de changer la description d'une condition. (**OfferCondition**)
- *setOfferConditionLogicalOperator* permet de changer l'attribut *_logicalOperator* d'une condition. (**OfferCondition**)

- *getOfferConditionOperations* permet de récupérer une liste des opérations associé à une condition. (**OfferOperationData**)
- *setOfferConditionOperationValue* permet de modifier la valeur d'une opération d'une condition d'une offre. (**OfferOperation**)
- *setOfferConditionOperationLogicalOperator* permet de modifier l'opérateur logique d'une opération d'une condition d'une offre. (**OfferOperation**)
- *setOfferConditionOperationRelationalOperator* permet de modifier l'opérateur relationnel d'une opération d'une condition d'une offre. (**OfferOperation**)
- *getReductionTypePossibilities* permet de récupérer une liste sous forme de tableau de **String** décrivant tous les différents types de réductions disponible.
- *getLogicalOperatorTypePossibilities* permet de récupérer une liste sous forme de tableau de **String** décrivant tous les différents types de d'opérateur de logique disponible.
- *getRelationalOperatorTypePossibilities* permet de récupérer une liste sous forme de tableau de **String** décrivant tous les différents types de d'opérateur de relations disponible.

1.3.1.2. **PlaceRateData**

La classe **PlaceRateData** est une class abstraite contenant les attributs et les getters nécessaire à la classe **PlaceRate**, grâce à cela cette classe ne contenant que des getters peut être utilisé par l'UI.

Elle possède 4 attributs:

- *_price* correspond à la valeur du prix de la place.
- *_id* correspond à l'identifiant de la place.
- *_listOffers* est une liste d'offres associé à la place.
- *_color* correspond à la couleur de la place.

Cette classe contient 4 methodes:

- *getPrice* permet de récupérer l'attribut *_price*.
- *getListOffers* permet de récupérer l'attribut *_listOffers*
- *getColor* permet de récupérer l'attribut *_color*.
- *getId* permet de récupérer l'attribut *_id*.

Le constructeur de la classe prend en paramètre l'identifiant de la place, une couleur, la valeur de la place.

1.3.1.3. **PlaceRate**

La classe **PlaceRate** gère le prix d'une place, sa couleur et ses offres. Elle hérite directement de **PlaceRateData** qui contient les attributs nécessaires .

Cette classe contient 4 methodes:

- *setPrice* permet de modifier l'attribut *_price*.
- *setColor* permet de modifier l'attribut *_color*.
- *addOffer* permet d'ajouter une offre dans *_listOffers*
- *removeOffer* permet de retirer une offre à la place

Le constructeur de la classe prend en paramètre l'identifiant de la place, une couleur, la valeur de la place.

1.3.1.4. **OfferRate**

La class **OfferData** est une class abstraite contenant les attributs et les getters nécessaire à la classe **Offer**, grâce à cela cette classe ne contenant aucun setters elle peut être utilisé par l'**UI**.

Elle possède 5 attributs:

- *_name* est le nom de l'offre
- *_reduction* correspond à la valeur de l'offre
- *_description* correspond à la description de l'offre
- *_listConditions* est une liste de **OfferCondition**
- *_reductionType* est le type de la réduction (**ReductionType**), par pourcentage ou par valeur.

Cette classe contient 5 méthodes:

- *getReductionTypeValue* permet de récupérer l'attribut *_reductionType* sous forme d'une **String**.
- *getOfferConditionsData* permet de récupérer l'attribut *_listConditions* sous forme d'une liste de **OfferConditionData**.
- *getDescription* permet de récupérer l'attribut *_description*.
- *getReduction* permet de récupérer l'attribut *_reduction*.
- *getName* permet de récupérer l'attribut *_name*.

1.3.1.5. **Offer**

La classe **Offer** permet de gérer les offres qui pourront être attribué à une place.

Elle possède 1 type:

- **ReductionType** correspond au type de la réduction, réduction avec pourcentage ou réduction avec valeur.

Cette classe contient 8 méthodes:

- *addCondition* permet d'ajouter une condition dans *_listConditions*.
- *removeCondition* permet de retirer une condition de la list *_listConditions*.
- *setReduction* permet de modifier l'attribut *_reduction*.
- *setName* permet de modifier l'attribut *_name*.
- *setDescription* permet de modifier l'attribut *_description*.
- *setReductionType* permet de modifier l'attribut *_reductionType*.
- *getOfferConditions* permet de récupérer l'attribut *_listConditions* sous forme d'une liste de **OfferCondition**.
- *getReductionType* permet de récupérer l'attribut *_reductionType*.

1.3.1.6. **OfferConditionData**

La class **OfferConditionData** est une class abstraite contenant les attributs et les getters nécessaire au bon fonctionnement de la classe **OfferCondition**, étant donnée que cette classe ne contient aucune méthode de modification elle peut être utilisé par l'**UI**.

Elle possède 4 attributs:

- *_name* correspond au nom de la condition.
- *_description* correspond à la description de la condition.
- *_listOfferOperations* est une liste d'**OfferOperation**.
- *_logicalOperator* est un enum de type **LogicalOperator**, il défini si la condition doit être validé indépendamment ou dépendant des autres conditions.

Cette classe contient 4 méthodes:

- *getName* permet de récupérer l'attribut *_name*

- *getLogicalOperatorValue* permet de récupérer l'attribut *_logicalOperator* sous forme de **String**.
- *getDescription* permet de récupérer l'attribut *_description*
- *getOfferOperationsData* permet de récupérer l'attribut *_listOfferOperations* sous forme d'une liste de **OfferOperationData**.

1.3.1.7. **OfferCondition**

La class **OfferCondition** permet de gérer les conditions d'une offre.

Cette classe contient 7 méthodes:

- *addOperation* permet d'ajouter une opération dans *_listOfferOperations*
- *removeOperation* permet de retirer une opération dans *_listOfferOperations*
- *setLogicalOperator* permet de modifier l'attribut *_logicalOperator*
- *setName* permet de modifier l'attribut *_name*
- *setDescription* permet de modifier l'attribut *_description*
- *getOfferOperations* permet de récupérer l'attribut *_listOfferOperations* sous forme d'une liste de **OfferOperation**.
- *getLogicalOperator* permet de récupérer l'attribut *_logicalOperator*

1.3.1.8. **OfferOperationData**

La class **OfferOperationData** est une class abstraite contenant les attributs et les getters nécessaire au bon fonctionnement de la classe **OfferOperation**, étant donnée que cette classe ne contient aucune méthode de modification elle peut être utilisé par l'UI.

Elle possède 3 attributs:

- *_value* est la valeur de l'opération qui permettra de valider ou non l'opération.
- *_relationalOperator* est un enum de type **LogicalOperator**, il déterminera l'opération devra être traité avec les autres opérations ou non.
- *_logicalOperator* est un enum de type **RelationalOperator**, il déterminera la façon dont la valeur devra être vérifié, supérieur ou égal à celle donné, inférieur etc.

Cette class est composé de 3 méthodes:

- *getValue* permet de récupérer l'attribut *_value*.
- *getLogicalOperatorValue* permet de récupérer l'attribut *_logicalOperator* sous forme de **String**.
- *getRelationalOperatorValue* permet de récupérer l'attribut *_relationalOperator* sous forme de **String**.

1.3.1.9. **OfferCondition**

La class **OfferOperation** permet de gérer une opération qui servira à la classe **OfferCondition**.

Cette class est composé de 5 méthodes:

- *setLogicalOperator* permet de modifier l'attribut *_logicalOperator*
- *setRelationalOperator* permet de modifier l'attribut *_relationalOperator*
- *setValue* permet de modifier l'attribut *_value*
- *getLogicalOperator* permet de récupérer l'attribut *_logicalOperator*
- *getRelationalOperator* permet de récupérer l'attribut *_relationalOperator*.

1.3.1.10. Operator

1.3.1.10.1. LogicalOperator

LogicalOperator est un enum, il permet de définir un opérateur logique dans une condition tel que le “>=”, “<”, “==”, “!=” etc.

1.3.1.10.2. RelationalOperator

RelationalOperator est un enum, il permet de définir un opérateur relationnel dans une condition tel que le “&”, “|”, “&&” etc.

1.3.2. Save

La classe **Save** permet de sauvegarder le plan de salle en cours de réalisation ou à la fin de sa réalisation.

La classe possède 1 attribut :

- *_currentPath* correspond au chemin pour accéder au fichier de sauvegarde.

Elle se compose de 4 méthodes :

1. *saveTo* permet de sauvegarder un objet à un endroit précis de l'ordinateur c'est-à-dire à l'emplacement du path désiré.
2. *loadFrom* permet de charger un fichier depuis un path et de retourner un objet.
3. *getCurrentPath* permet de récupérer le path.
4. *setCurrentPath* permet de modifier le path.

6. Conclusion

Après réalisation de notre projet, nous avons pu identifier les points forts de celui-ci. Tout d'abord, nous avons mis l'accent sur le design de l'interface qui est épuré, simple et moderne et nous avons intégré de nombreux raccourcis. Ceci permet d'améliorer l'expérience utilisateur. De plus, le calcul des prix automatiques est réalisable en fonction de trois paramètres : le prix minimum, le prix maximum et le revenu total augmentant ainsi les possibilités des prix pour l'utilisateur. Nous avons aussi implémenté un undo/redo ne se limitant pas à 10 actions. Grâce au Pattern Command, notre limite dépend alors de la ram mais nous avons quand même fixé une limite à 10000. Afin de rendre notre projet exportable et utilisable dans de plus nombreuses situations, il est possible de réaliser un export en JSON. Enfin, lors de la réalisation de ce projet, nous avons décidé d'implémenter des services tels que le prix, la sauvegarde pouvant ainsi être réutilisé dans d'autres projets. Nous avons aussi réalisé un moteur graphique indépendant du projet et de ce fait, réutilisable dans d'autres projets nécessitant un moteur graphique.

Par la suite, nous avons identifié les points faibles de notre projet. Nous avons alors remarqué qu'il existait de légers bugs dans certaines situations bien précises et rares qui pourraient être corrigés dans le futur.

Pour conclure, notre application peut être commercialisable. En effet, celle-ci permet d'exporter au format JSON toutes les informations de la Room avec ses propres informations, la scène, les multiples sections, les prix, les offres... Ainsi, une application tierce serait capable de réutiliser ces informations à leur guise.

De plus, nous avons réalisé une plateforme web. Il est possible d'y charger un plan de salle afin de mettre en vente les places. Nous avons donc implémenté un système de paiement et un système de connexion afin de rendre notre projet commercialisable. Cette plateforme est accessible en ligne à l'url suivante <https://innoevent.ga/>. Pour s'y connecter, nous avons créé un utilisateur de test avec comme nom d'utilisateur root@root.com et le mot de passe root. Et pour

tester le paiement en ligne, vous pouvez utiliser une carte bleue de test qui est 5555 5555 5555 4444 et 02/22.

7. Contribution des membres de l'équipe

- Rémi Gastaldi :
 - Implémentation de l'engine
- Léo Hubert :
 - Implémentation des prix et des offres
 - Réalisation des popups, sidebars, design
 - Réalisation du site web
- Maud Marel :
 - Implémentation de la room
 - Réalisation des sidebars, design
 - Rédaction du rapport final
- Khaled Nasri :
 - Réalisation de l'algorithme de génération automatique des sièges
 - Réalisation de l'algorithme de génération des prix en fonction d'un prix minimum, d'un prix maximum et d'un revenu total