

Louis THIDET

Université Gustave-Eiffel



Rapport de projet

Eiffel Tutoring Solutions - GustaveTutorService

TABLE DES MATIÈRES

PRÉSENTATION DU PROJET.....	4
Structure des services.....	4
Fonctionnalités des projets.....	4
Classes des projets.....	5
FONCTIONNEMENT ET LIMITES.....	7
Structure des données.....	7
1. Etudiant.....	7
2. Tuteur.....	7
3. Séance de tutorat.....	7
4. Message.....	8
Fonctionnement des réservations.....	8
Limites.....	9
INSTALLATION DES SERVICES WEB.....	10
I. Importation des projets des services dans l'environnement où ils vont être mis en place.....	10
II. Mise en place du serveur où vont être lancés les services web.....	11
III. Installation du service web Bank.....	11
IV. Installation du service web GustaveTutorService.....	12
1. On commence par définir le projet GustaveTutorService comme étant client du service web Bank.....	12
2. Ensuite, on crée le service web GustaveTutorService.....	12
MANUEL D'UTILISATION.....	14
Gestion des services.....	14
1. Lancement de EiffelTutoringService.....	14
3. Consultation des logs.....	16
Connexion aux services.....	17
1. Inscription.....	17
2. Connexion.....	17

Guide du tuteur.....	18
1. Ajouter une séance de tutorat.....	18
2. Valider une prise de séance.....	19
3. Consulter et annuler les séances de tutorat.....	20
Guide de l'étudiant.....	21
1. Ajouter des fonds (Étudiants extérieurs uniquement).....	21
2. Réserver une leçon.....	22
3. Consulter et annuler les séances de tutorat.....	23

PRÉSENTATION DU PROJET

Le projet se décompose en deux parties, la partie RMI et la partie Services Web. La première est constituée des deux projets `Eclipse EiffelTutoringSolutionsClient` et `EiffelTutoringSolutionsServer`, alors que la seconde est constituée des projets `Eclipse Bank`, `GustaveTutorService` et `GustaveTutorServiceClient`. La première partie est dédiée aux utilisateurs de l'Université Gustave-Eiffel alors que la seconde est dédiée aux étudiants extérieurs à l'université.

Structure des services

La partie RMI représente un service appelé `EiffelTutoringSolutions`. Plutôt que de séparer notre partie RMI en un client destiné aux étudiants et un autre destiné aux tuteurs, on a décidé que tuteurs et étudiants accéderaient à un seul client se connectant au serveur du service `EiffelTutoringSolutions`. On a décidé cela pour deux raisons :

- D'abord, car cela permet à plusieurs tuteurs d'ajouter des cours en même temps. S'il en avait été autrement, il aurait fallu que soit les étudiants soit les tuteurs hébergent le service, ce qui n'aurait laissé la possibilité qu'à un tuteur ou un étudiant à la fois d'utiliser le service, puisque le service n'est hébergeable et ne doit être hébergé que sur une seule JVM.
- Ensuite parce que cela permet d'intégrer sur le côté serveur du service une interface graphique, qu'un administrateur peut consulter pour modifier ou consulter facilement les données du service, si cela est nécessaire.

La partie Service Web représente un service appelé `GustaveTutorService`. Elle est composée de deux services web : `GustaveTutorService`, et `Bank`, qui permet de convertir des devises en euro en utilisant l'API du convertisseur de devises `Fxtop.com`. `GustaveTutorService` est à la fois un client du service web `Bank` et du service RMI `EiffelTutoringSolutions`. Les étudiants extérieurs à l'université accèdent au service `GustaveTutorService` via le projet client de `GustaveTutorService`, qui est `GustaveTutorServiceClient`.

Fonctionnalités des projets

`EiffelTutoringService`

- Hébergement de la partie RMI et dépendance de la partie web
- hébergement de la base de données des services
- interface graphique d'administration
- ajout, suppression et modification des données
- Création des logs et consultation de ceux-ci

EiffelTutoringServiceClient

- Client de EiffelTutoringService
- Interface graphique d'utilisateur
- Inscription et connexion des utilisateurs, étudiants et tuteurs
- Ajout, et réservation des séances de tutorat
- Calendrier intégré pour visualiser les séances

Bank

- Client du service web Fxtop pour obtenir la conversion d'une devise en euro

GustaveTutorService

- Client de la partie RMI, pour permettre d'accéder aux méthodes de EiffelTutoringService sur le web
- Client du service web Bank

GustaveTutorServiceClient

- Client de GustaveTutorService
- Fonctionnalités similaires à EiffelTutoringServiceClient mais uniquement pour les étudiants
- Interface graphique d'utilisateur alternative pour permettre d'ajouter les fonds nécessaires au paiement des séances de tutorat

Classes des projets

EiffelTutoringService

- Un package gui contient la classe main du programme, laquelle est une JFrame de la librairie Swing, et contient l'interface graphique.
- Un package managers, qui contient la classe clientManager, et son interface IClientManager, qui servent au projet serveur à communiquer avec le projet client, puis les classes CSVManager (Gestion des fichiers de données), LogManager (Gestion des logs) et SessionManager (Gestion des sessions client).

EiffelTutoringServiceClient

- Un package gui contient la classe main du programme, laquelle est une JFrame de Swing, puis différentes classes étendant JPanel, appelées dans main (ConnexionPanel, SignUpPanel, LogInPanel, StudentPanel, TutorPanel), et CalendarView, qui affiche un fichier calendrier.html dans une WebView de JavaFX.
- Un package managers qui comprend IClientManager une classe CSVManager (dont les méthodes et fonctions diffèrent du CSVManager du projet serveur).

Bank

- Un package main, qui comprend Bank, une classe convertissant une somme d'une certaine devise en euro, via un autre package : Fxtop API.

GustaveTutorService

- Un package main, qui comprend GustaveTutorService et les classes liées à Bank, puis
- un package managers, qui contient IClientManager.

GustaveTutorServiceClient

- Un package gui similaire celui d'EiffelTutoringServiceClient, mais aux classes modifiées et sans TutorPanel.
- Un package main qui contient les classes liées à GustaveTutorService
- Un package managers qui contient IclientManager.

FONCTIONNEMENT ET LIMITES

Structure des données

Les services reposent sur quatre structures de données : les étudiants, les tuteurs, les séances de tutorat et les messages (contenues respectivement dans `students.csv`, `tutors.csv`, `lessons.csv` et `messages.csv`). Chacune de ces quatre structures de données contient une colonne dédiée à l'identification d'une observation (`student_id`, `tutor_id`, `lesson_id`, `message_id`). Quand une nouvelle observation apparaît dans une structure, elle prend en identifiant la valeur de la dernière observation ajoutée, incrémentée de 1.

1. Etudiant

L'étudiant comprend les variables suivantes :

- **name**
- **surname**
- **email**
- **password**
- **is_gustave** : informe de si l'étudiant est externe ou interne à l'Université Gustave-Eiffel
- **balance** : fonds dont dispose l'étudiant externe pour prendre des séances de tutorat
- **student_id**

2. Tuteur

L'étudiant comprend les variables suivantes :

- **name**
- **surname**
- **email**
- **password**
- **tutor_id**

3. Séance de tutorat

La séance de tutorat comprend les variables suivantes :

- **date** : format YYYY-MM-DD
- **start_time** : heure de début (format HH:MM)
- **end_time** : heure de fin (format HH:MM)
- **is_online** : est-ce que la séance est en présentiel ou non
- **type** : type de la séance, parmi ceux proposés par le service
- **cost**
- **student_id** : ID de l'étudiant ayant réservé la séance
- **tutor_id** : ID du tuteur ayant créé la séance
- **lesson_id**

4. Message

Le message comprend les variables suivantes :

- **waiting_for_answer** : indique si le message attend une réponse ou non
- **answer** : état de la réponse au message (*no* par défaut)
- **recipient_id** : ID de la personne recevant le message (soit l'étudiant, soit le tuteur)
- **student_id**
- **tutor_id**
- **lesson_id**
- **message_id**

Fonctionnement des réservations

Lorsqu'une séance est créée, son `student_id` par défaut est *free*. Il devient *pending* lorsqu'un étudiant souhaite réserver la séance, et il prend l'ID de l'étudiant si le tuteur de la séance valide sa réservation (ou redevient *free*, s'il la refuse). Afin d'éviter les conflits, il n'est pas possible à un étudiant de sélectionner une séance et de la réserver si un autre étudiant l'a déjà demandée en réservation.

La réservation des séances fonctionne par le biais des messages. Quand un étudiant souhaite réserver une séance, non seulement la variable `student_id` de celle-ci passe sur *pending*, mais un message est créé. Ce message récupérera l'ID de la séance, l'ID du tuteur ayant créé la séance et l'ID de l'étudiant souhaitant la réserver. Il aura aussi l'ID du tuteur dans `recipient_id`, et sa variable `waiting_for_answer` contiendra *yes*.

Quand un utilisateur a son ID dans la variable `recipient_id` d'un message, il voit ce message s'afficher sur son interface graphique. Si ce message a pour valeur *yes* dans `waiting_for_answer`, cela signifie qu'il attend une réponse, autrement, il s'agit simplement d'un message informatif, que l'utilisateur a le choix de supprimer ou non (S'il le supprime, il disparaît de la base de données).

Le message attendant une réponse apparaît pour le tuteur uniquement : quand un étudiant souhaite lui réserver une séance, tandis que le message informatif apparaît pour le tuteur et pour l'étudiant. Pour le tuteur, ce dernier apparaît quand un étudiant annule une séance validée, et pour l'étudiant, il apparaît quand le tuteur supprime la séance, ou quand il refuse de la lui accorder.

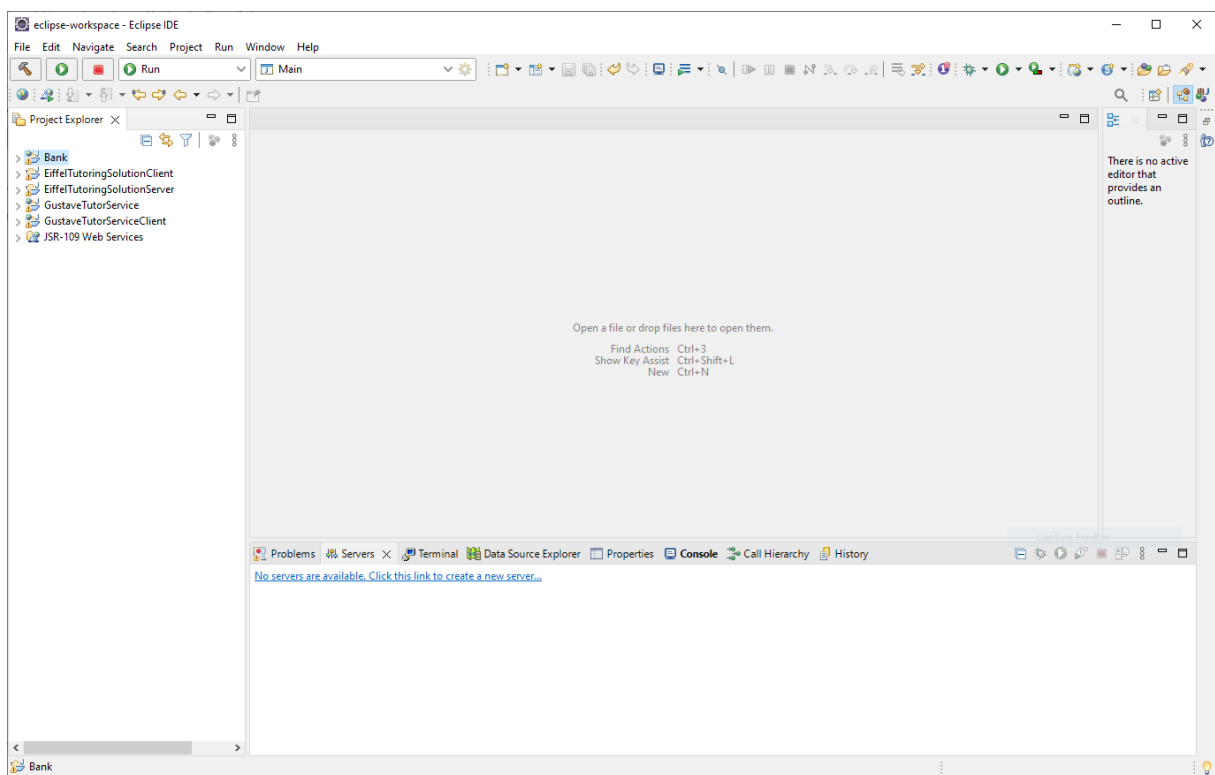
Limites

- Pour une raison que nous ne sommes pas parvenu à identifier, lorsque le serveur sur lequel ont été intégrés les services web a été fermé puis rouvert plus tard, des erreurs liées à la désérialisations apparaissent (*Found character data inside an array element while deserializing*). A chaque fois que le serveur ferme, il est donc nécessaire de réinstaller les services web à sa prochaine ouverture.
- Pour améliorer l'interface graphique du client, nous avons cherché à intégrer un calendrier, sur lequel seraient affichées les leçons du client (étudiant ou tuteur). Etant donné que nous n'en avons pas trouvé de librairie adéquate pour créer ce composant, nous avons intégré dans notre projet un fichier html, par le biais de la classe WebView, comprise dans la librairie JavaFX. Dans ce fichier html se trouve du Javascript et la librairie Javascript FullCalendar, qui permet d'afficher un calendrier tel que nous le souhaitons.
- Nous ne nous sommes pas bien rendu compte du temps dont nous étions doté pour réaliser le projet, et avons intégré des fonctionnalités au détriment de la propreté du code. Malgré un temps conséquent investi, et le fait que les services soient fonctionnels, il demeure quelques problèmes. Le code, parfois, est peu lisible, et un peu anarchique ; il comprend des fonctions dispensables ou peuvent être raccourcies qui demeurent ; et il comporte sans doute des failles que l'utilisateur mal avisé pourrait exploiter, si celui-ci disposait du code du programme client et en réécrivait certaines parties. On a néanmoins essayé d'intégrer des sécurités, via un système de session, qui empêche l'utilisateur mal connecté de faire ce qu'il souhaite.
- Etant donné que notre connaissance du Javascript est assez primaire, nous ne sommes pas parvenus à directement charger dans le fichier html des données issues de Java. Pour charger les données dans le fichier html, nous avons fait en sorte qu'à chaque fois que le client charge ses leçons via Java, elles soient écrites localement, sur son ordinateur, dans des fichiers csv.
- Quand un élève s'inscrit sur EiffelTutoringSolutions, donc le service interne à l'Université Gustave-Eiffel, il est censé recevoir sur l'adresse qu'il a entrée pour s'inscrire un lien de confirmation pour activer son compte. Pour des raisons évidentes, nous n'avons pas pu intégrer cette fonctionnalité dans le logiciel.
- Quand l'utilisateur de GustaveTutorService souhaite ajouter des fonds à son compte, il peut en ajouter autant qu'il le souhaite, puisque le logiciel n'est pas connecté à de vrais moyens de paiement.

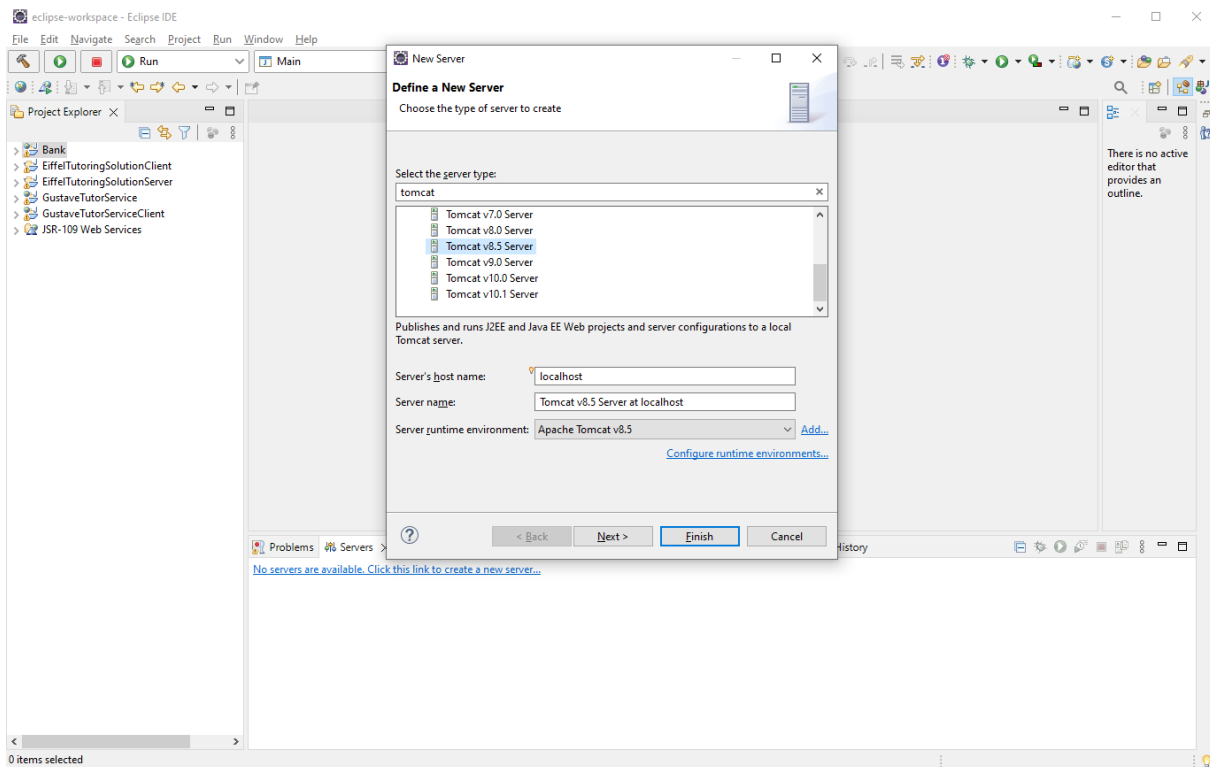
INSTALLATION DES SERVICES WEB

Afin de pouvoir utiliser non seulement Eiffel Tutoring Service mais GustaveTutorService, il est nécessaire au préalable de disposer d'un serveur sur lequel sont installés les services web GustaveTutorService et Bank. Les étapes nécessaires à cette installation sont illustrées par des captures d'écran. Nous utilisons dans notre exemple un serveur local avec l'IDE Eclipse.

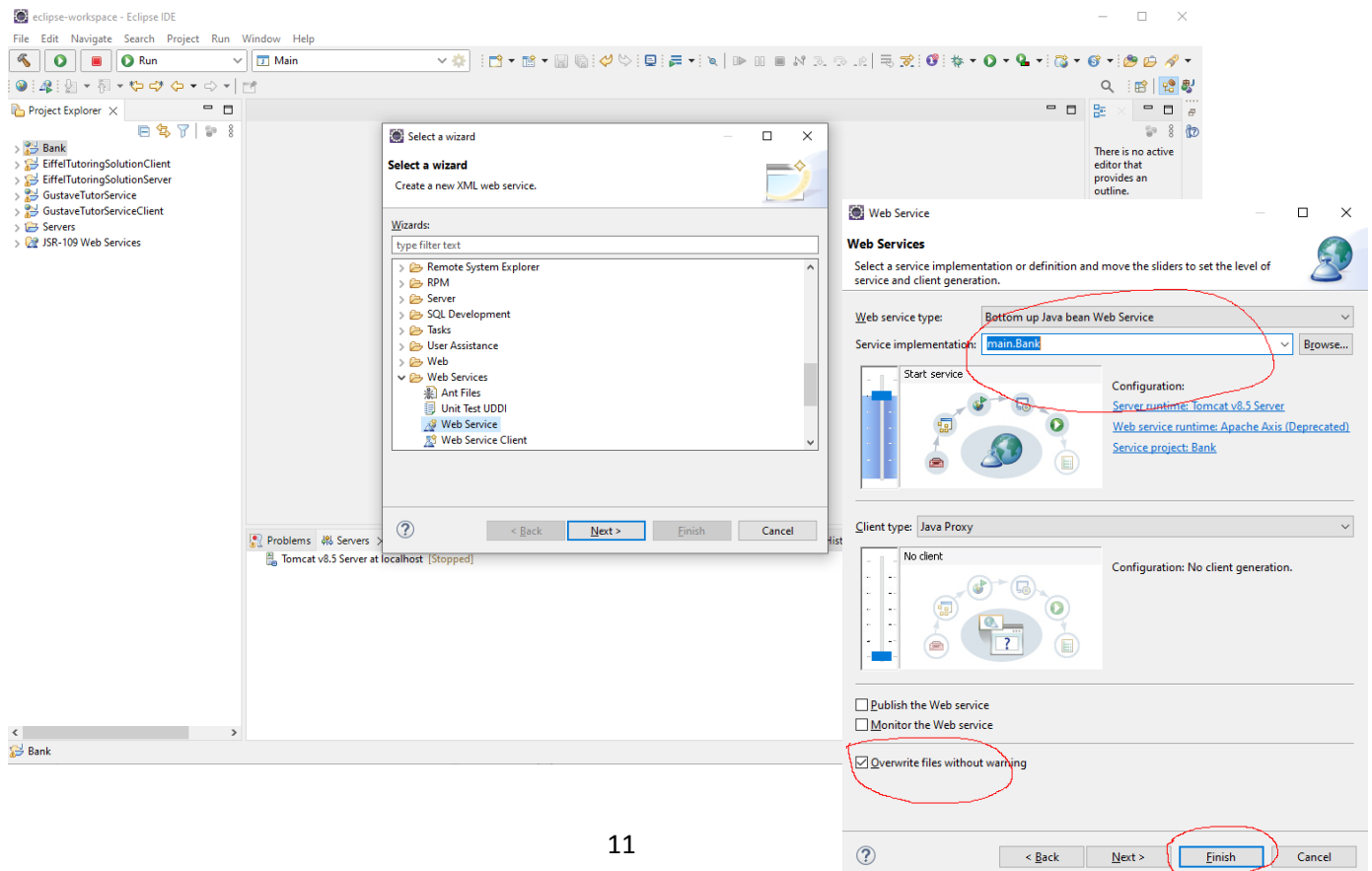
I. Importation des projets des services dans l'environnement où ils vont être mis en place



II. Mise en place du serveur où vont être lancés les services web

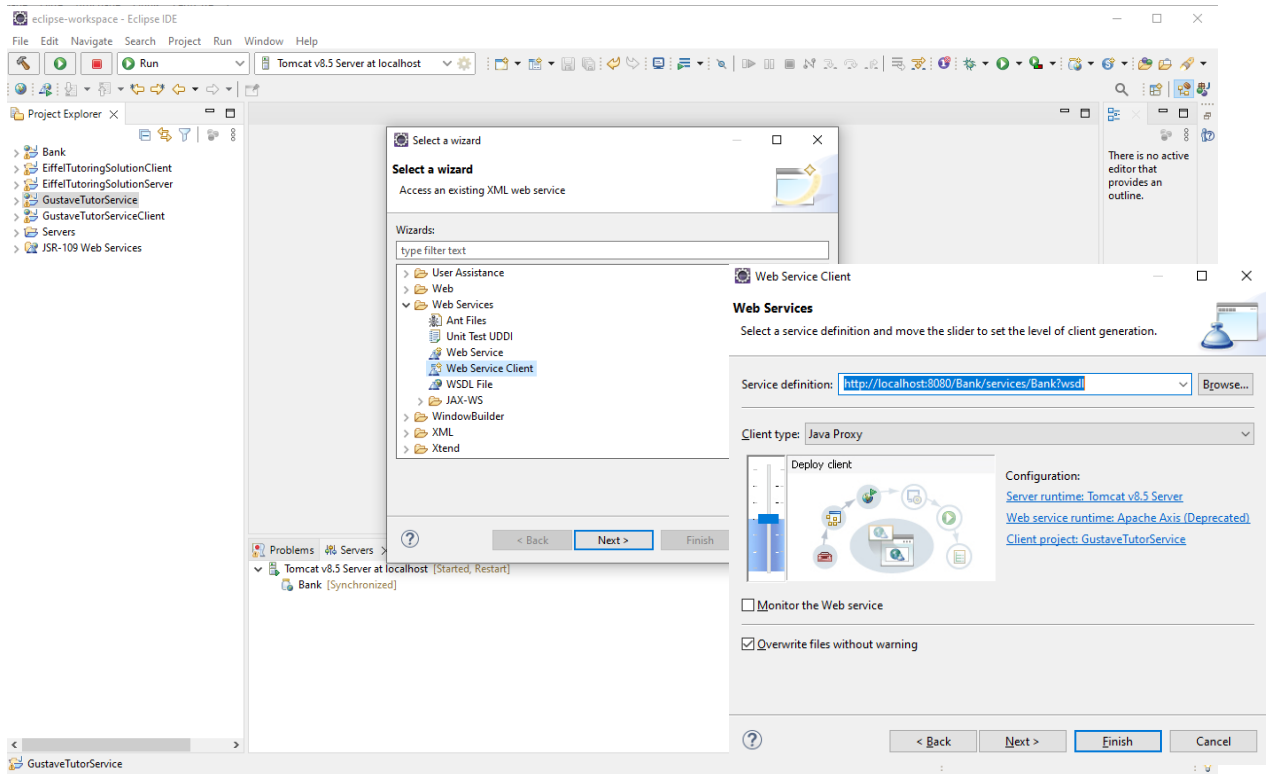


III. Installation du service web Bank

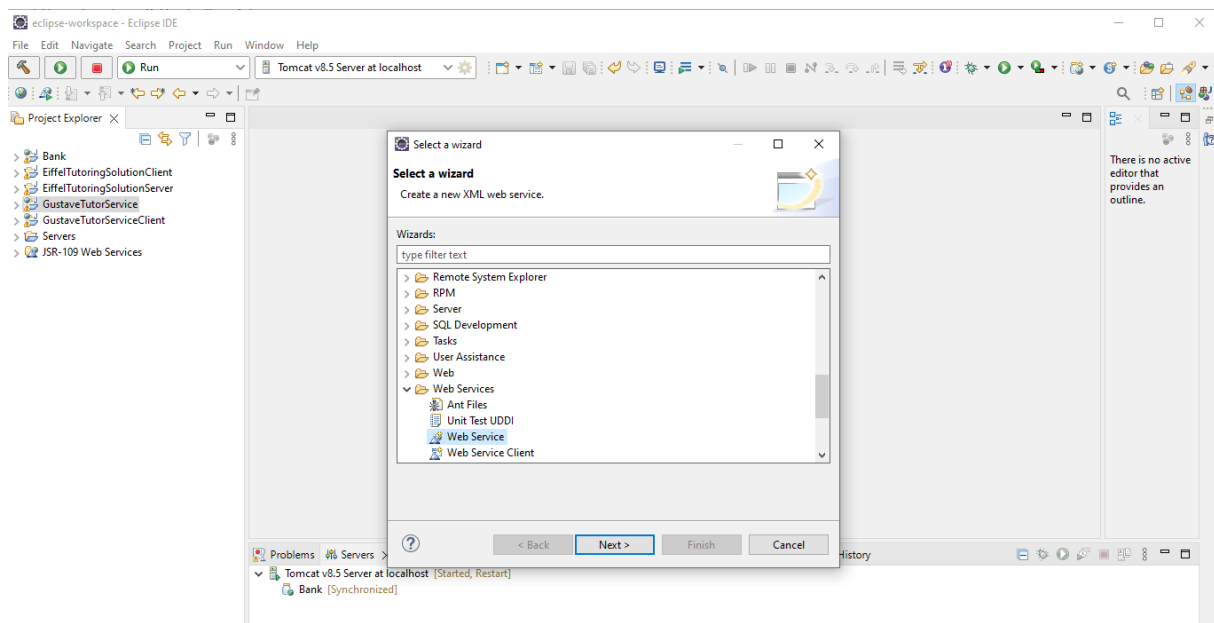


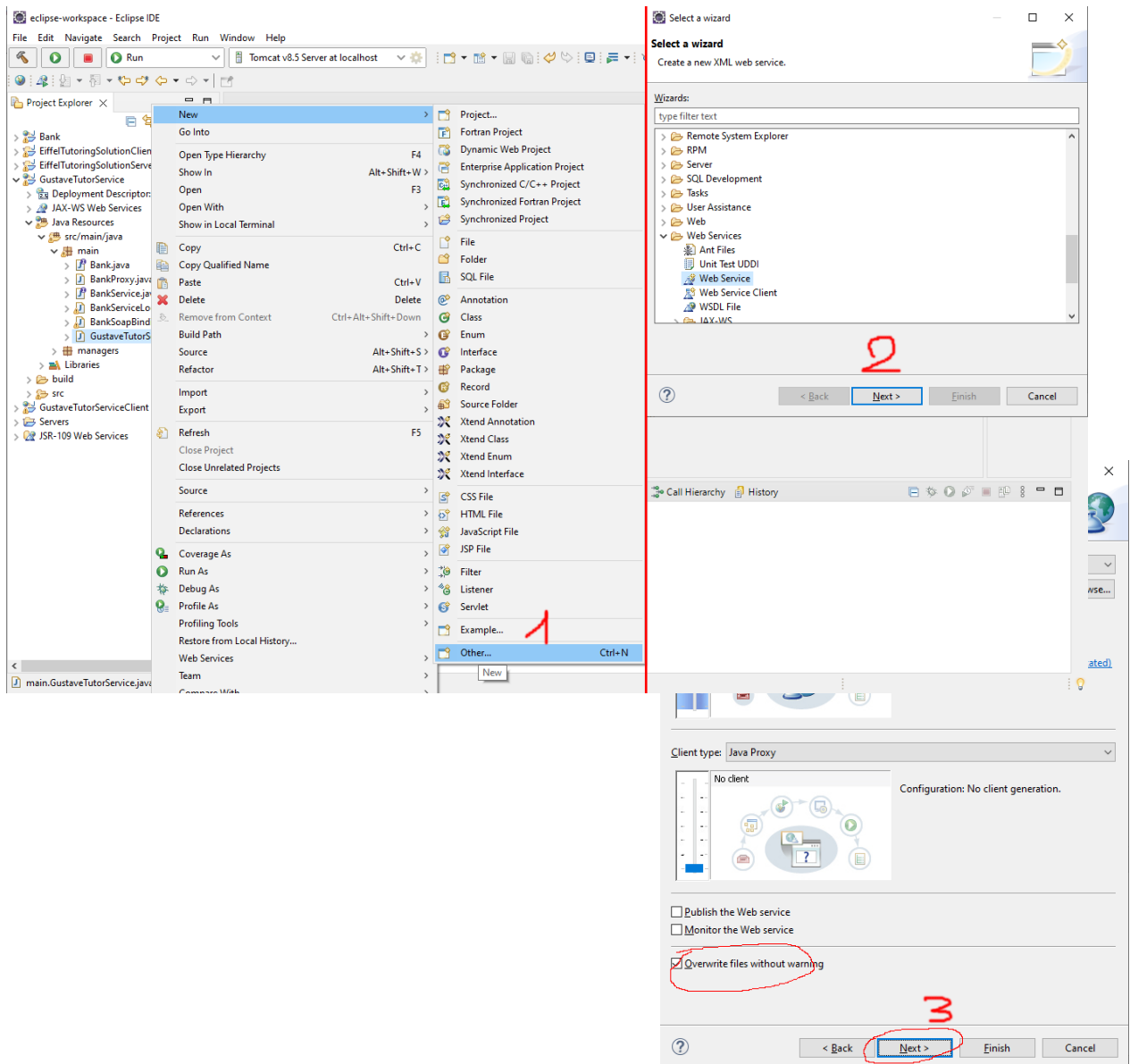
IV. Installation du service web GustaveTutorService

1. On commence par définir le projet GustaveTutorService comme étant client du service web Bank.



2. Ensuite, on crée le service web GustaveTutorService



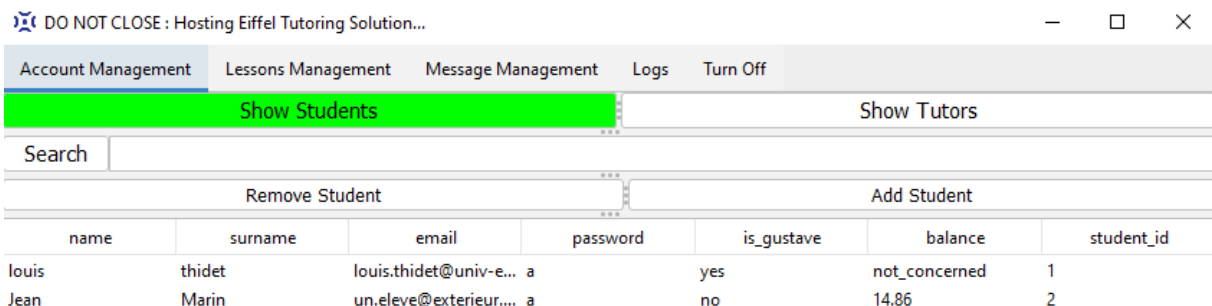
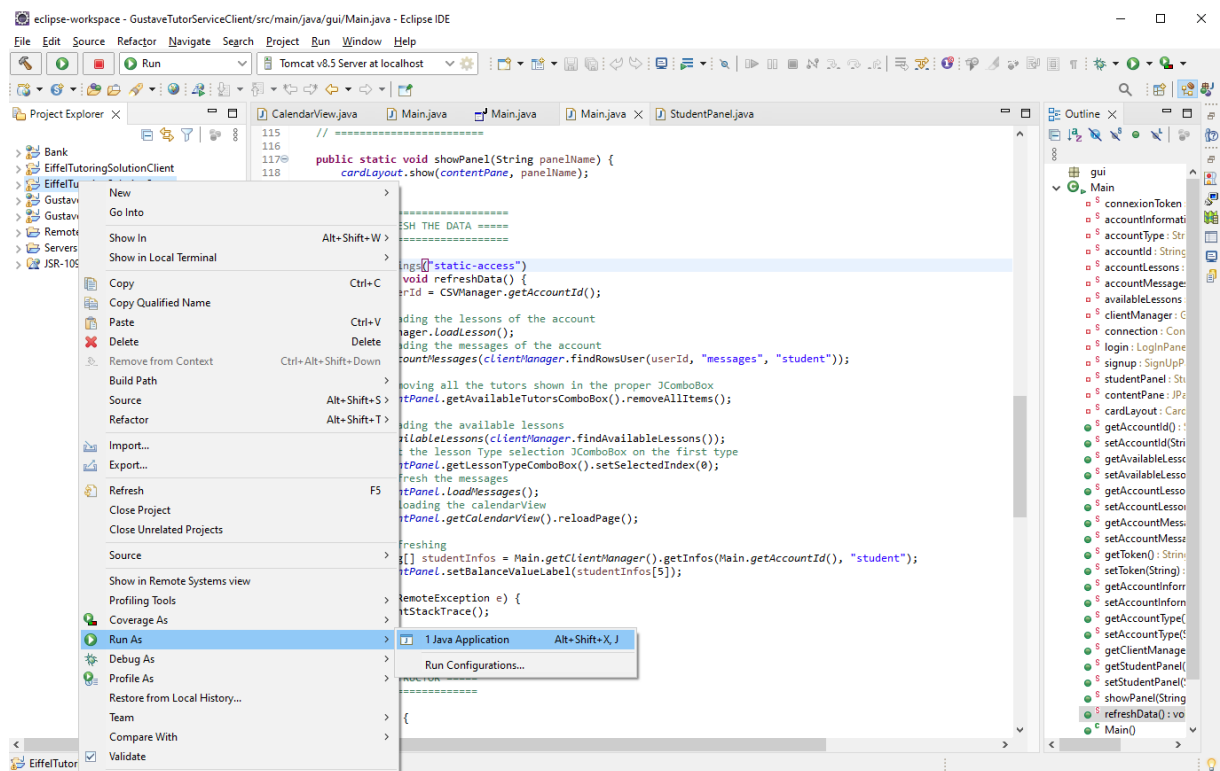


MANUEL D'UTILISATION

Gestion des services

1. Lancement de EiffelTutoringService

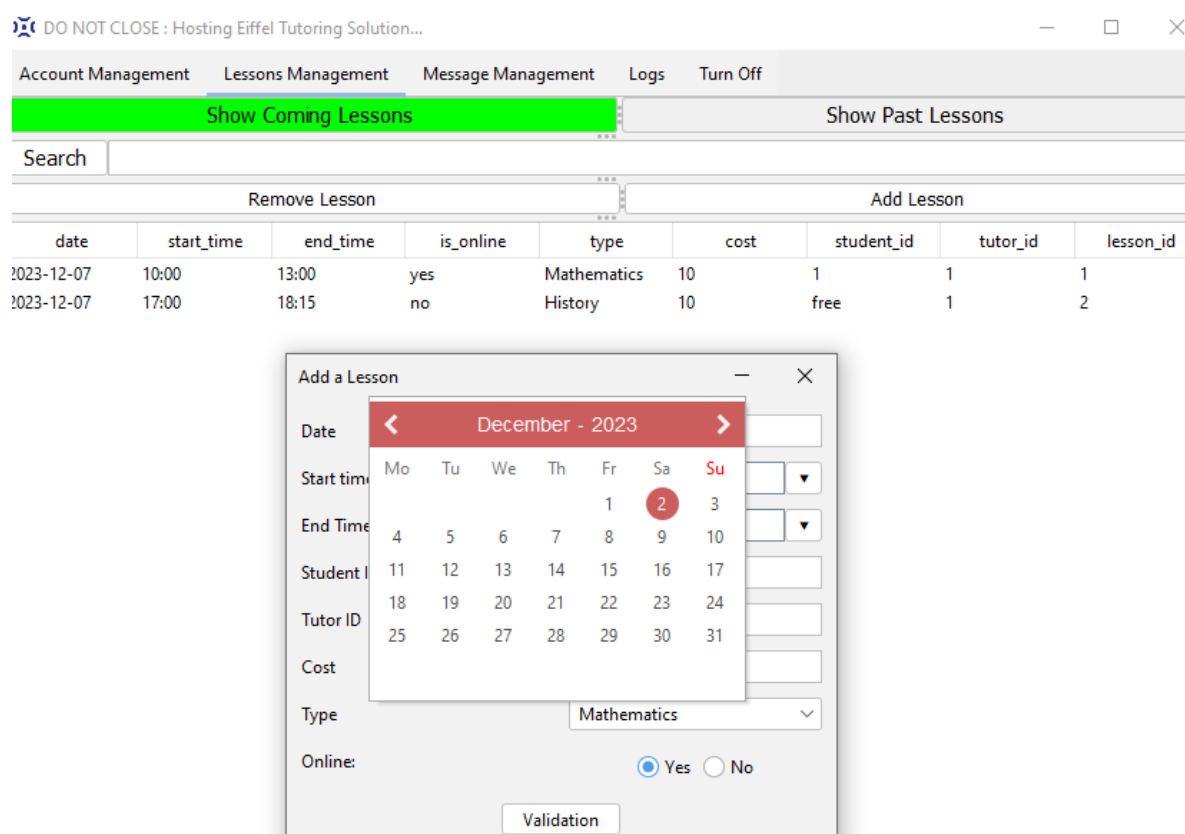
Une fois les services web installés, l'intégralité des services sont prêts à être utilisés. Pour commencer à utiliser GustaveTutorService et Eiffel Tutoring Solution, il faut lancer le projet EiffelTutoringSolutionServer.



Le service hébergeur est prêt à recevoir des clients.

2. Recherche, ajout et modification des données via l'interface graphique

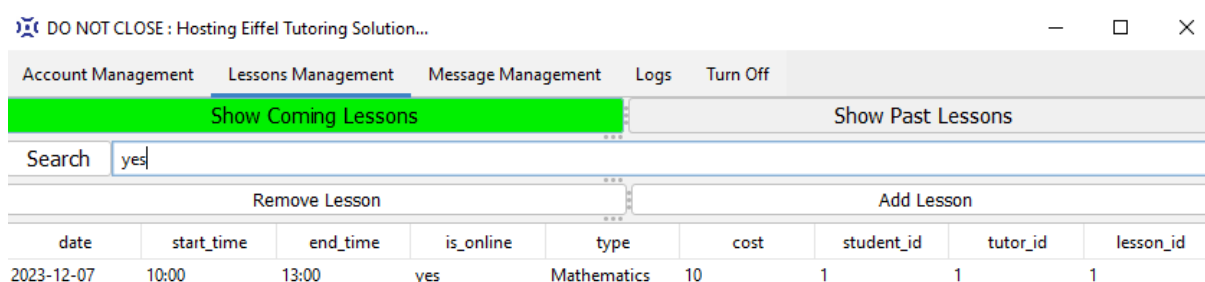
L'administrateur peut accéder aux données du service en se rendant dans EiffelTutoringService/src/data/, ou alors utiliser l'interface graphique de EiffelTutoringService, qui permet de visualiser, rechercher, ajouter, supprimer ou modifier des données.



The screenshot shows the 'Eiffel Tutoring Solution' interface with the 'Lessons Management' tab selected. The 'Show Coming Lessons' button is highlighted in green. Below the tabs, there is a search bar and a table of lessons. The 'Add Lesson' dialog box is open, showing a calendar for December 2023 with the 2nd highlighted. The dialog also includes fields for Start time, End Time, Student ID, Tutor ID, Cost, Type (set to Mathematics), and Online status (set to Yes).

date	start_time	end_time	is_online	type	cost	student_id	tutor_id	lesson_id
2023-12-07	10:00	13:00	yes	Mathematics	10	1	1	1
2023-12-07	17:00	18:15	no	History	10	free	1	2

L'administrateur ajoute une leçon à la base de données.



The screenshot shows the same interface as before, but with the search bar containing the text 'yes'. The table now displays only the lesson with 'is_online' set to 'yes'.

date	start_time	end_time	is_online	type	cost	student_id	tutor_id	lesson_id
2023-12-07	10:00	13:00	yes	Mathematics	10	1	1	1

L'administrateur recherche les séances de tutorat dispensées en ligne.

DO NOT CLOSE : Hosting Eiffel Tutoring Solution...

Account ManagementLessons ManagementMessage ManagementLogsTurn Off

Search

Remove Message

Add Message

waiting_for_answer	answer	recipient_id	student_id	tutor_id	lesson_id	message_id
yes	yes	1	1	1	1	1
no	no	1	2	1	2	2
yes	yes	1	2	1	2	3
no	no	1	2	1	2	4

Confirmation

Answer's state?

Yes No Pending

L'administrateur change la réponse d'un message reçu par le tuteur dont l'ID est 1

3. Consultation des logs

L'utilisateur peut voir en direct ses propres actions ou ceux des clients des services (sous réserve que la fonction `infoMessage()` ait été implantée dans la fonction qui traduit l'opération qu'ils accomplissent; `infoMessage()` est une fonction qui affiche une ligne dans l'onglet Logs du projet hébergeur et l'enregistre dans un fichier `logs.txt`, sur l'ordinateur lançant le projet hébergeur).

DO NOT CLOSE : Hosting Eiffel Tutoring Solution...

Account Management	Lessons Management	Message Management	Logs	Turn Off
2023-12-02 00:41:39: Service has been Turned On 2023-12-02 01:02:50: A user has established a connection to the service 2023-12-02 01:03:02: Student: Jean Marin (ID: 2) has connected (Token: vqmfnn15576bzyf) 2023-12-02 01:04:00: A user has established a connection to the service 2023-12-02 01:04:18: Tutor: super prof (ID: 1) has connected (Token: q2u73kofda8amb9) 2023-12-02 01:05:28: Student Jean Marin (ID:2) has deleted a message (ID: 5)				

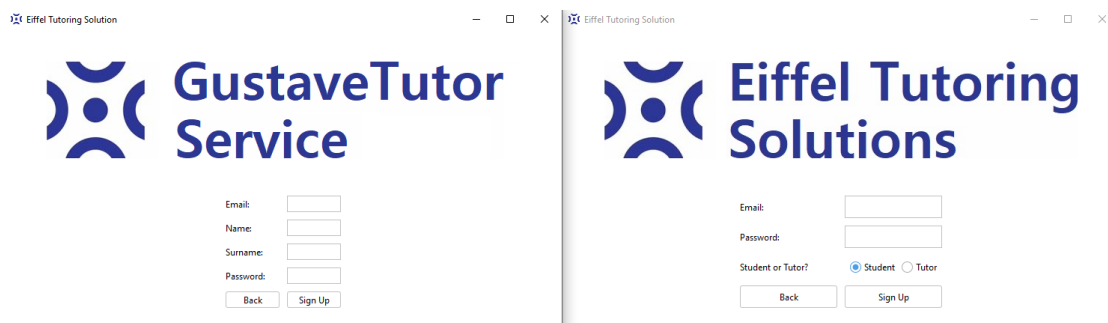
Empty Log File Show Log File in Explorer

L'administrateur consulte les logs du service.

Connexion aux services

1. Inscription

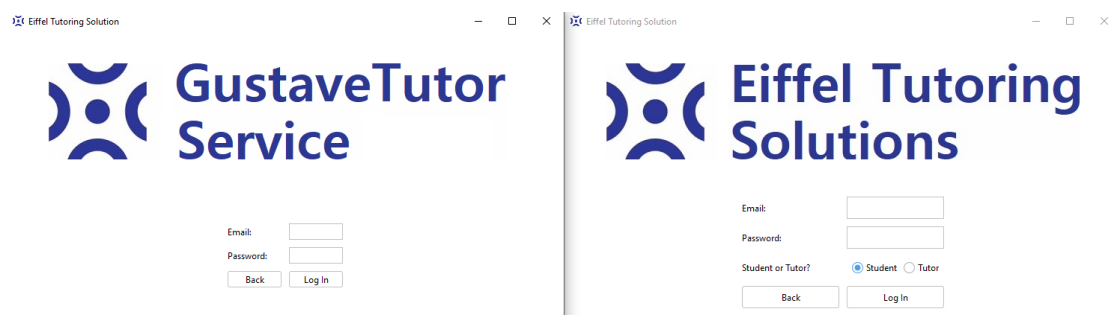
Pour s'inscrire sur un service, l'utilisateur n'a qu'à cliquer sur le bouton Sign In qui s'affiche à sur l'écran de démarrage (ConnectionPanel dans les fichiers). L'inscription sur EiffelTutoringSolutions et GustaveTutorService diffère un peu, puisque GustaveTutorService ne concerne que des étudiants, et des étudiants pouvant venir de n'importe où, alors que EiffelTutoringSolutions concerne les étudiants et les professeurs de l'Université Gustave-Eiffel. Pour cette raison, lorsqu'un utilisateur s'inscrit sur GustaveTutorService, il doit entrer son nom et son prénom. Quand une personne est utilisatrice de EiffelTutoringSolutions, et par conséquent est interne à l'Université Gustave-Eiffel, elle ne les entre pas, parce que l'on a l'assurance que l'adresse email qu'elle renseigne est liée à son nom et son prénom. Pour s'inscrire et puis se connecter, l'utilisateur d'EiffelTutoringSolutions doit entrer une adresse email se terminant par @univ-eiffel.fr ou @edu.univ-eiffel.fr.



The image shows two side-by-side browser windows displaying registration forms. The left window is for 'GustaveTutor Service' and the right is for 'Eiffel Tutoring Solutions'. Both forms have a logo at the top and input fields for Email, Name, Surname, and Password. The GustaveTutorService form has a 'Sign Up' button, while the Eiffel Tutoring Solutions form has a 'Sign Up' button and a 'Student or Tutor?' section with radio buttons for 'Student' (selected) and 'Tutor'. Both forms also have a 'Back' button.

2. Connexion

Pour les deux services, la connexion nécessite d'entrer l'adresse email et le mot de passe. Mais les utilisateurs d'EiffelTutoringService doivent aussi informer le logiciel de s'ils sont étudiant ou tuteur.

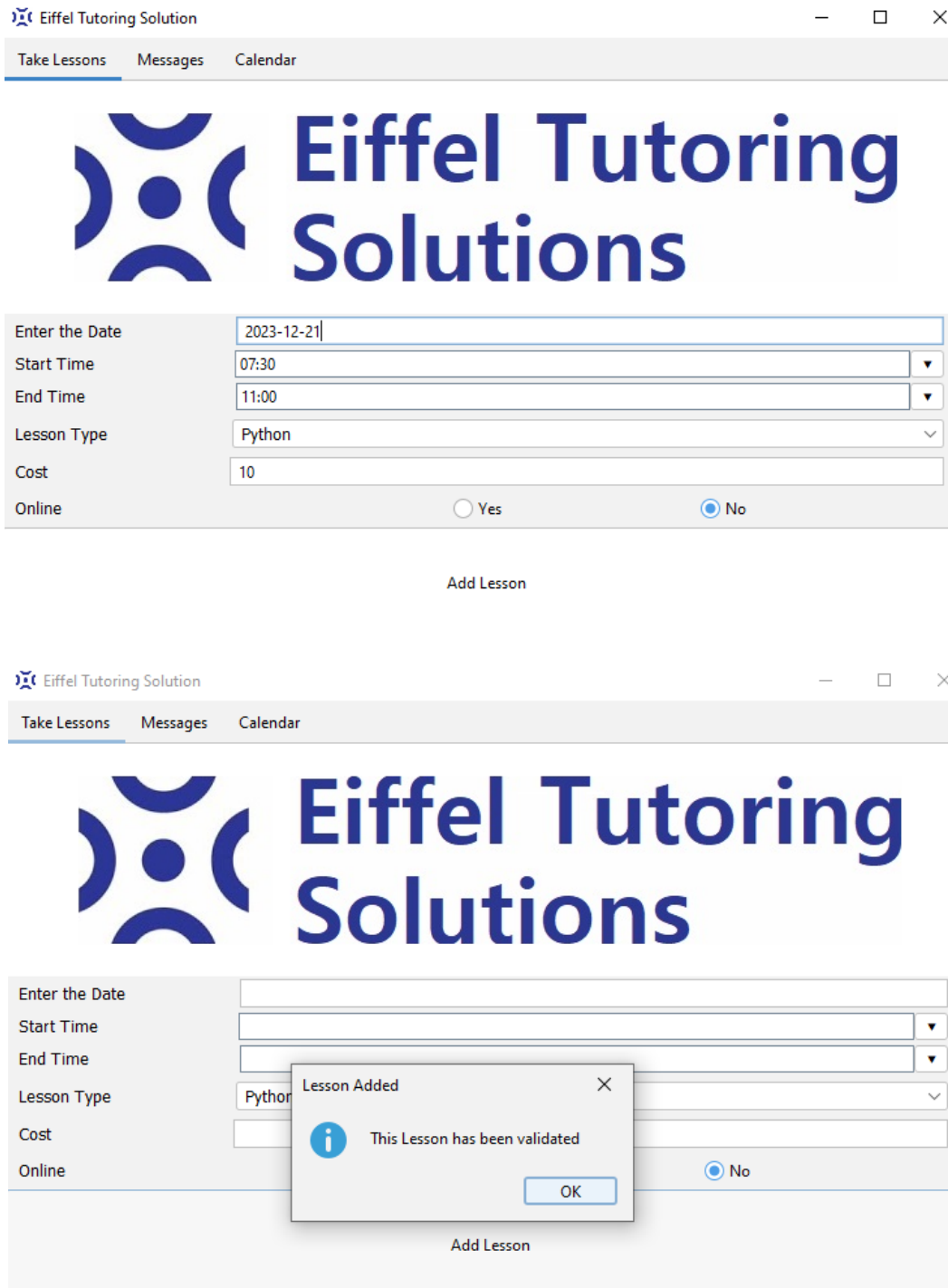


The image shows two side-by-side browser windows displaying login forms. The left window is for 'GustaveTutor Service' and the right is for 'Eiffel Tutoring Solutions'. Both forms have a logo at the top and input fields for Email and Password. The GustaveTutorService form has a 'Log In' button, while the Eiffel Tutoring Solutions form has a 'Log In' button and a 'Student or Tutor?' section with radio buttons for 'Student' (selected) and 'Tutor'. Both forms also have a 'Back' button.

Guide du tuteur

1. Ajouter une séance de tutorat

Une fois connecté, le tuteur peut ajouter une leçon dans le premier onglet de son interface.



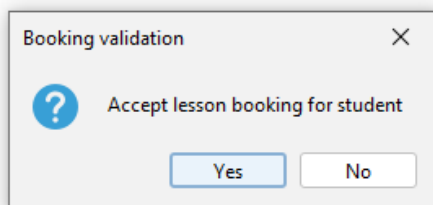
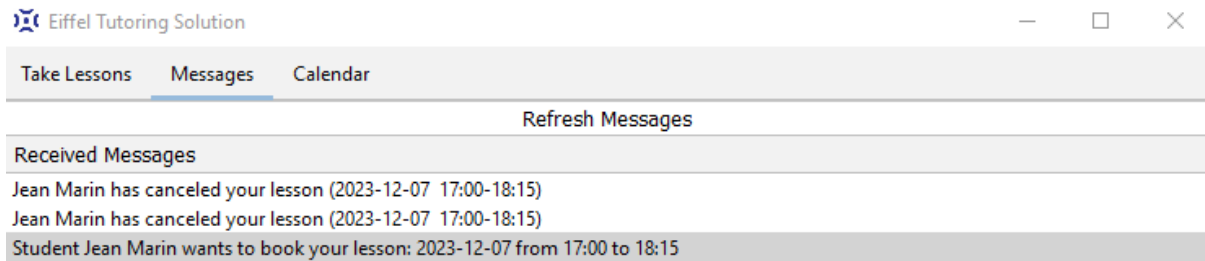
The screenshot shows the 'Eiffel Tutoring Solution' interface with the 'Take Lessons' tab selected. The form contains the following fields and options:

- Enter the Date: 2023-12-21
- Start Time: 07:30
- End Time: 11:00
- Lesson Type: Python
- Cost: 10
- Online: ☐ Yes ☒ No

Below the form is an 'Add Lesson' button. A second screenshot shows the same interface with a 'Lesson Added' dialog box displayed, indicating that the lesson has been validated. The dialog box contains an information icon, the text 'This Lesson has been validated', and an 'OK' button.

2. Valider une prise de séance

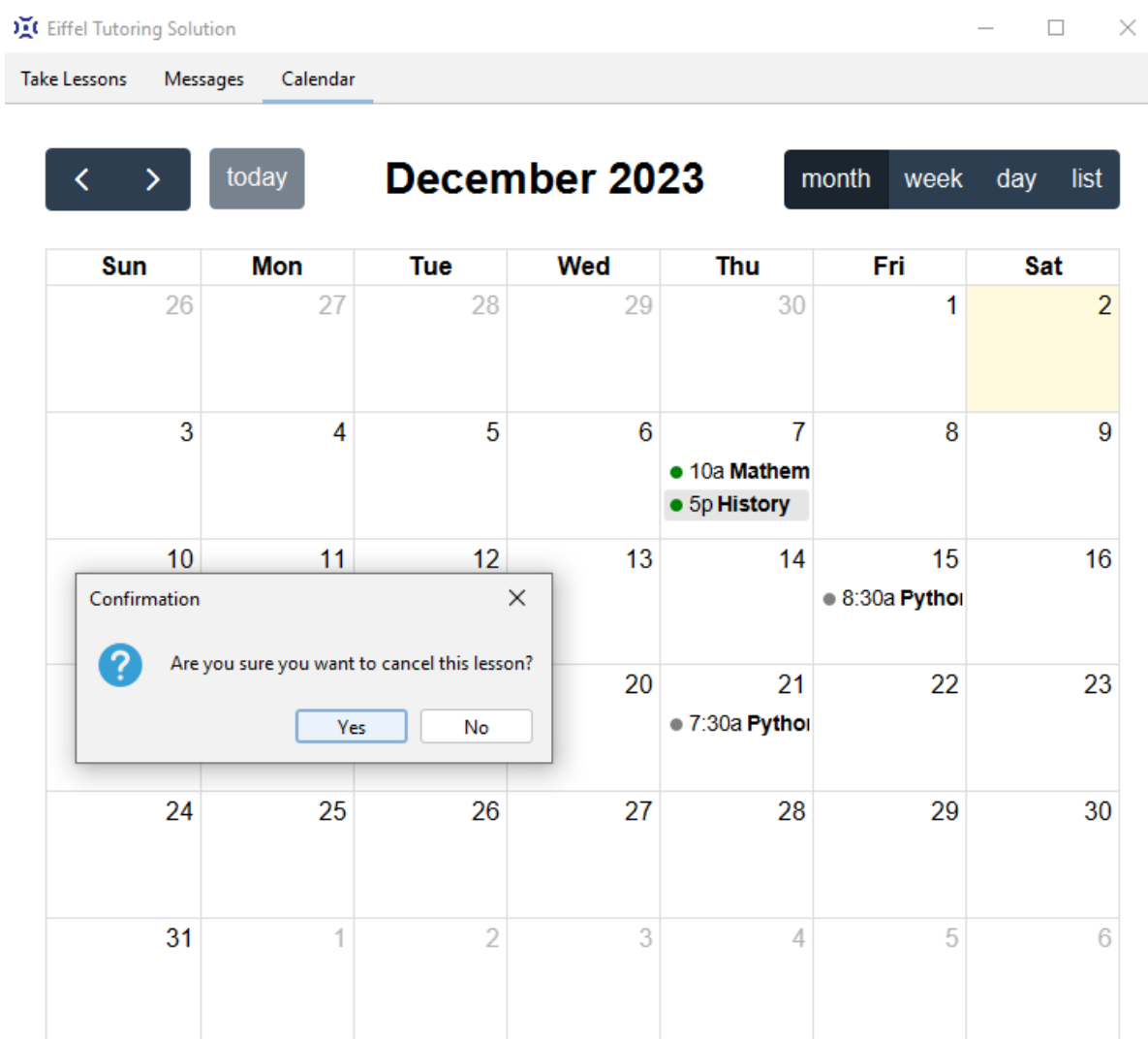
Lorsqu'un étudiant a demandé au tuteur l'accès à sa séance, le tuteur n'a plus qu'à la valider.



Un tuteur accepte de prendre l'étudiant Jean Marin pour une séance de tutorat.

3. Consulter et annuler les séances de tutorat

Le tuteur peut voir les séances de tutorat qu'il a entrées dans le service dans un calendrier. Si elles sont prises par un étudiant, elles sont affichées en vert, si elles sont en attente d'être prises, elles sont en orange, et si elles n'ont pas été demandées, elles sont en gris. Lorsque le tuteur annule une séance, elle est supprimée, et si un étudiant l'avait prise, alors il est remboursé. Le tuteur n'a pas la possibilité d'annuler la prise de séance d'un étudiant s'il l'a acceptée.



The screenshot shows the 'Eiffel Tutoring Solution' interface. At the top, there are tabs for 'Take Lessons', 'Messages', and 'Calendar'. The 'Calendar' tab is active, displaying a monthly view for December 2023. The calendar grid shows dates from 26 to 6. A confirmation dialog box is overlaid on the calendar, asking 'Are you sure you want to cancel this lesson?' with 'Yes' and 'No' buttons. The dialog box has a blue question mark icon and a close button (X). The calendar also shows several lessons: '10a Mathem' and '5p History' on Thursday the 7th, '8:30a Python' on Friday the 15th, and '7:30a Python' on Friday the 21st. The date 2nd is highlighted in yellow.

Un tuteur sur le point d'annuler l'une de ses séances de tutorat qu'un élève a réservée.

Guide de l'étudiant


1. Ajouter des fonds (Étudiants extérieurs uniquement)

Si un étudiant utilise le service GustaveTutorService, il doit ajouter des fonds à son compte, afin de pouvoir réserver des séances de tutorat (sauf si le professeur qui les donne a décidé de les dispenser gratuitement). Pour cela, il dispose d'un onglet Balance, où il peut choisir d'ajouter un certain montant de la devise de son choix, qui sera converti en euros.



Eiffel Tutoring Solution

Take Lessons Messages Calendar **Balance**

 **GustaveTutor
Service**

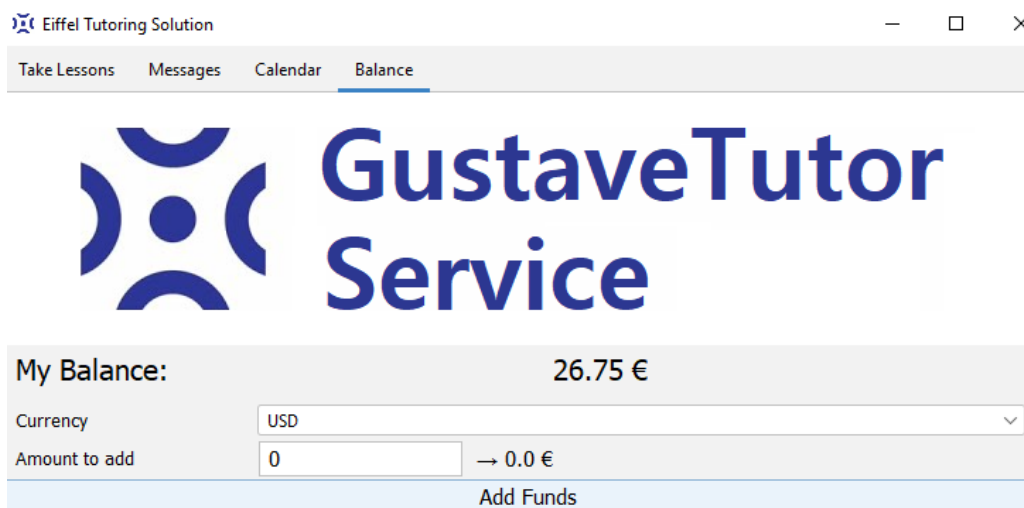
My Balance: 14.86 €

Currency: USD

Amount to add: 12 → 11.891784758695866 €


Add Funds

[Modify payment methods](#)



Eiffel Tutoring Solution

Take Lessons Messages Calendar **Balance**

 **GustaveTutor
Service**

My Balance: 26.75 €

Currency: USD

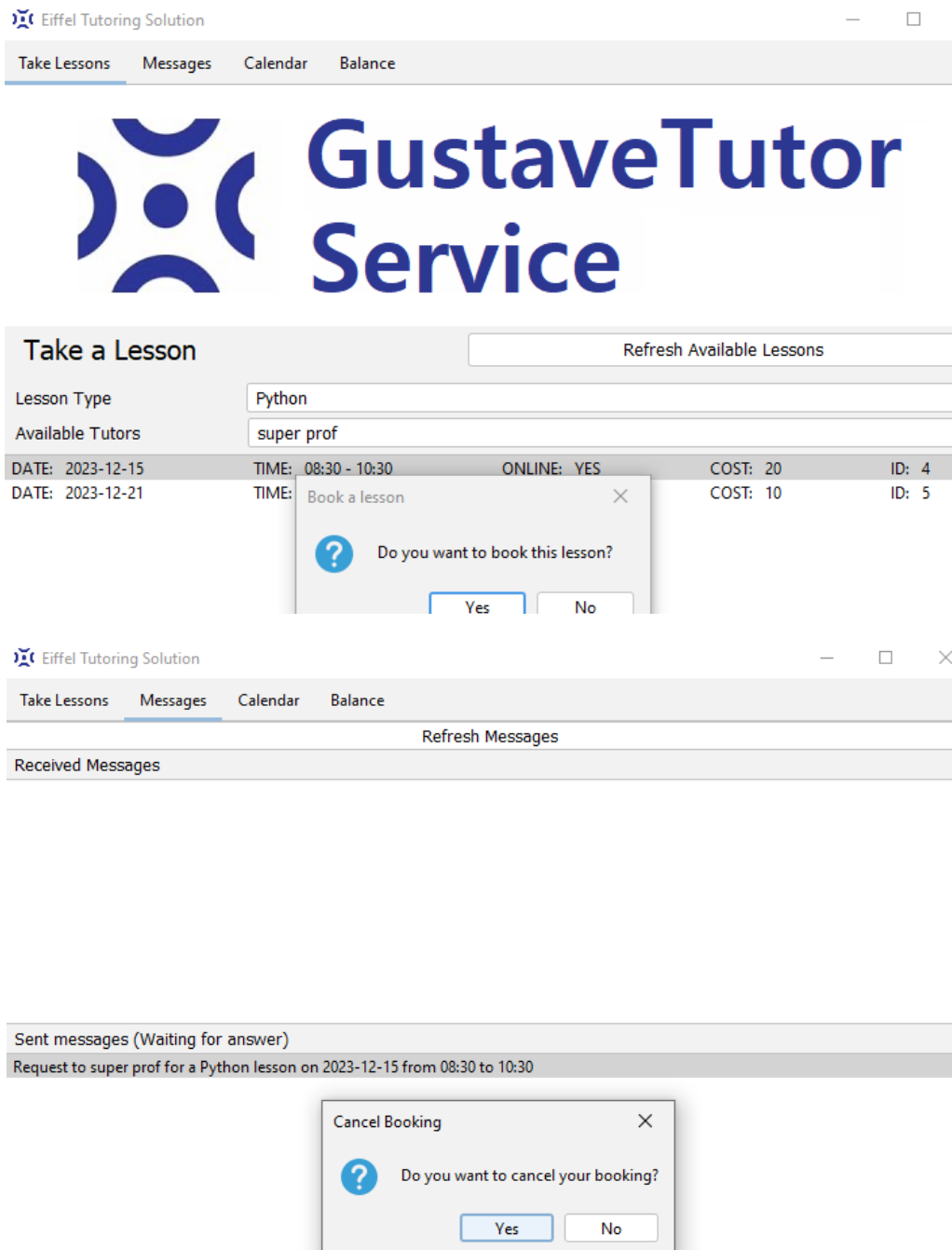
Amount to add: 0 → 0.0 €

Add Funds

[Modify payment methods](#)

2. Réserver une leçon

Pour réserver une leçon, l'étudiant doit la sélectionner en double-cliquant dessus dans l'onglet Take Lesson. Il choisit le tuteur qu'il souhaite pour la matière de son choix puis il clique sur l'une des séances disponibles. Quand l'utilisateur a demandé de réserver la séance, il peut annuler sa requête dans l'onglet Message, dans la liste Sent Messages. Et si la requête est validée, un message de confirmation apparaît dans Received Messages.



The screenshot displays the 'Eiffel Tutoring Solution' web application. The top navigation bar includes 'Take Lessons', 'Messages', 'Calendar', and 'Balance'. The main header features the 'GustaveTutor Service' logo.

Take a Lesson Section:

- Lesson Type:** Python
- Available Tutors:** super prof
- Available Sessions:**

DATE	TIME	ONLINE	COST	ID
2023-12-15	08:30 - 10:30	YES	20	4
2023-12-21			10	5

A modal dialog 'Book a lesson' is shown with the question 'Do you want to book this lesson?' and 'Yes'/'No' buttons.

Messages Section:

- Received Messages:** (Empty)
- Sent messages (Waiting for answer):**

Request to super prof for a Python lesson on 2023-12-15 from 08:30 to 10:30

A modal dialog 'Cancel Booking' is shown with the question 'Do you want to cancel your booking?' and 'Yes'/'No' buttons.

3. Consulter et annuler les séances de tutorat

L'élève peut annuler une séance de tutorat de la même manière qu'un tuteur. Quand celle-ci sera annulée, elle sera alors libérée, et un autre étudiant pourra la réserver.

Eiffel Tutoring Solution — □ ×

Take Lessons Messages **Calendar** Balance

<
>
today
December 2023
month
week
day
list

Sun	Mon	Tue	Wed	Thu	Fri	Sat
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

Confirmation ×

? Are you sure you want to cancel this lesson?

Yes No

3:30a Python

Un étudiant sur le point d'annuler la séance de tutorat qu'il a réservée.