ERGANEO

L'AVENIR EST FAIT D'AUDACE

Cahier des charges mission UX/UI et développement

Argument Theory

16 décembre 2019

01 44 23 21 50 | connect@erganeo.com
37 rue de Lyon - 75012 Paris CS 32707
RCS de Paris 539 868 224

ERGANEO

L'AVENIR EST FAIT D'AUDACE

Sommaire

1. A propos d'Argument Theory

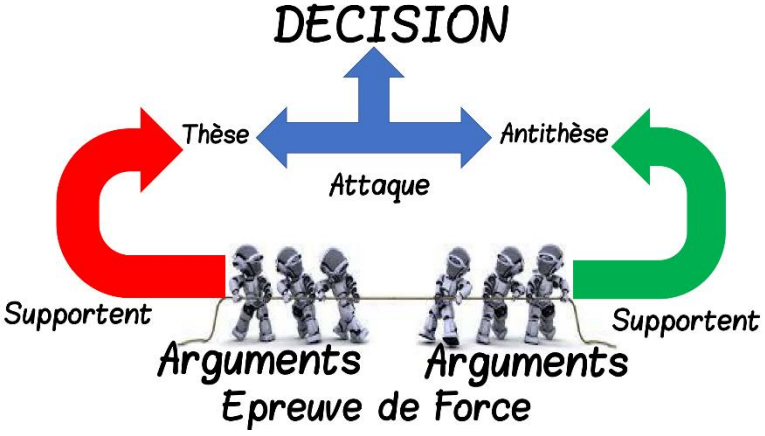
1.1 Ce que nous faisons

Argument Theory met à la disposition de ses clients sa plateforme d'Intelligence Artificielle, fondée sur l'argumentation computationnelle, pour résoudre des problèmes de décisions complexes et difficiles. Elle fournit une aide décisionnelle importante et éventuellement des solutions entièrement automatisées quand cela est nécessaire, en toute transparence et justification. Nous pouvons encoder rapidement et efficacement les politiques décisionnelles de nos clients dans différents domaines (par exemple la gestion automatisée de conformité, la gestion automatisée d'accès aux données, le trading, l'évaluation et la mitigation automatique de risques) et fournir les réponses dont ils ont besoin tout en expliquant pourquoi ces décisions sont prises et sur quelles données elles sont fondées. Cela peut alors aider le client à exploiter davantage les résultats.

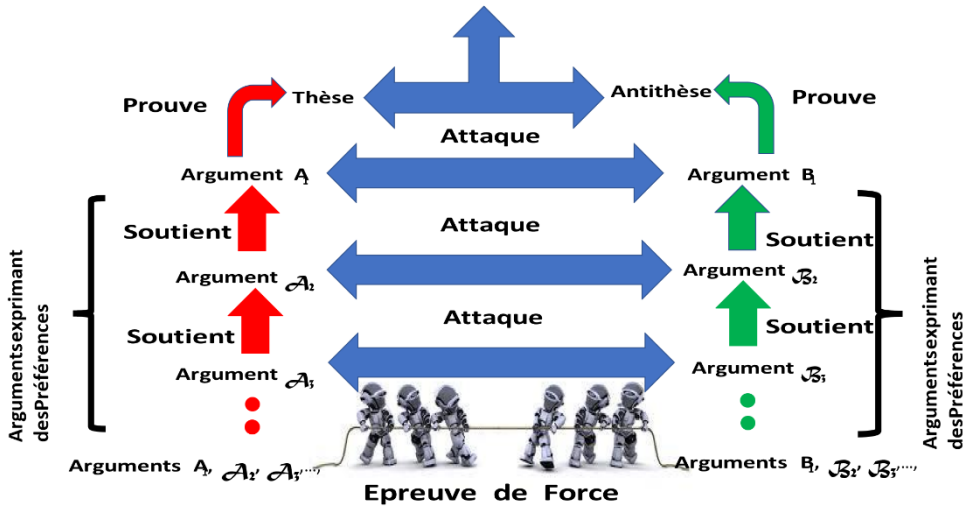
1.2 Notre technologie

Argument Theory est une plateforme SaaS d'Intelligence Artificielle (IA). Elle permet la prise en compte des exigences de problèmes de décisions impliquant de nombreux facteurs, éventuellement changeants et incomplets, et affectant de manière complexe et difficile la solution parmi une multitude de choix décisionnels. Notre approche repose sur l'argumentation computationnelle, une technologie qui permet à un système d'IA de disposer de capacités de raisonnement et de comportement similaires à celles d'un humain. Notre plateforme a en particulier la capacité d'évaluer les arguments pour ou contre une décision et de fournir des explications sur les raisons et les hypothèses à l'appui de cette décision. En outre, la technologie permet une adaptation modulaire et aisée de ses solutions et systèmes à des exigences nouvelles et changeantes.

Comment se prend une décision en argumentation computationnelle :



Dans le détail :



2. Éléments généraux

2.1 Objet du marché

2.1.1 Allotissement

Le présent marché comporte deux lots :

> Lot 1 : réalisation UX / UI
> Lot 2 : réalisation de la plateforme SaaS contenant l'applicatif Argument Theory

Lot 1 :

- Réalisation des écrans UX illustrant les parcours décrits dans le présent cahier des charges
- Réalisation de l'UI des différents écrans UX sur la base de l'identité graphique Argument Theory en cours de réalisation (marché séparé)
- Réalisation du CSS/HTML associé aux écrans UI

Lot 2 :

- Rédaction des spécifications fonctionnelles détaillées sur la base des éléments techniques fournis dans le présent cahier des charges
- Développement de la plateforme SaaS accueillant la solution Argument Theory
- Offre de maintenance corrective et évolutive (périmètre, garanties associées)

Annexes
Sont transmis aux candidats les annexes suivantes :

> Annexe 1 : AT_Gorgias_API_Doc, fichier PDF
> Annexe 2 : AT_Authoring_System, fichier PPTX

2.1.2 Type de marché

Marché de services à procédure adaptée selon l'article 30 du Décret n° 2016-360 du 25 mars 2016 relatif aux marchés publics.

2.1.3 Acheteur

SATT Erganeo
Société par actions simplifiée au capital de 1 000 000 euros.
Siège social : 37 rue de Lyon, CS 32707, 75012 Paris.
RCS : Paris B 539 868 224

2.1.4 Planning

La présente consultation est envoyée aux candidats par Erganeo le 12 décembre 2019.
La date limite de réception des offres est le jeudi 09 janvier 2020 à 12h.
Le choix des candidats se fera le mardi 14 janvier 2020 à 12h.
La réalisation des lots 1 et 2 est prévue dans un planning de 4 mois (avril 2020). Erganeo et Argument Theory sont à l'écoute des candidats quant à leurs propositions en matière de planning.

2.1.5 Contacts

Pour les aspects fonctionnels et techniques :
Pavlos MORAITIS
Téléphone : 01 76 53 02 85 / 06 83 22 20 66
Mail : pavlos@mi.parisdescartes.fr, pavlos@argument-theory.com

Pour les aspects administratifs :
Estelle VIDAL
Portable : 07 61 86 77 56
Mail : estelle.vidal@erganeo.com

2.1.6 Propriété intellectuelle

Aux termes des dispositions de code de propriété intellectuelle, Erganeo sera titulaire de l'ensemble des droits de propriété intellectuelle sur les productions réalisées dans le cadre de cette prestation, le co-contractant cédant, en contrepartie de la rémunération perçue au titre de la prestation, à Erganeo, tous droits y relatifs, et notamment sans que ceci soit limitatif les droits de reproduction, de représentation, d'adaptation, de traduction, de modification. Le co-contractant garantit que tous les contenus nécessaires à la réalisation des Productions définis dans le présent document, ne contiennent aucun élément qui puisse porter atteinte aux droits d'un tiers ou être constitutif de contrefaçon ou d'un acte de concurrence déloyale ou parasitaire. En outre, il garantit détenir l'ensemble des droits et autorisations nécessaires sur lesdits éléments. Le co-contractant ne peut en aucun cas faire usage de ces

Productions que ce soit à titre gratuit ou onéreux, sauf après autorisation écrite d'Erganeo. Le cas échéant, les images proposées par le co-contractant devront être libres de droits. Le co-contractant devra vérifier avant toute utilisation d'image que les droits nécessaires à la bonne exploitation de l'image sont acquittés auprès de son auteur et obtenir, le cas échéant les autorisations de diffusion.

Il est rappelé aux candidats qu'ils sont liés à Erganeo par un NDA et qu'il leur est strictement interdit de mentionner cette consultation ainsi que son contenu.

2.2 Fonctionnement de l'application

Gorgias propose, via web, des systèmes d'argumentation computationnelle que les utilisateurs finaux peuvent exploiter afin d'obtenir des réponses déduites par les conditions définies en amont et les paramètres personnalisés qui peuvent être saisis directement par les utilisateurs finaux. On appelle système un ensemble de paramètres qui, exécutés, amènent à la production d'une réponse.

Gorgias est le moteur d'intelligence artificielle développé par AT sur lequel reposent toutes les mises en œuvre concrètes présentées dans ce cahier des charges.

Actuellement une couche applicative web permet l'exploitation de Gorgias pour construite des scénarii ainsi que de consulter le résultat de leur exécution.

Au sein de la plateforme, l'interface de création de systèmes, dite interface d'authoring, permettra de créer des scenarios ou d'accéder aux scénarios existants pour les exécuter et les tester. L'interface de création de système permet également la création de la table des scenarii, grâce à elle, les développeurs de systèmes ont la capacité de construire de manière simple et efficace la table de scenarii.

A terme, la nouvelle plateforme développée à l'issue de cette consultation comprendra également des dashboards qui permettront la consultation du display des résultats du système de création de scenarii (authoring system) et vérifier, ainsi, son fonctionnement. Ces écrans de contrôle font partie de la consultation. Les écrans qui permettront d'exploiter les systèmes fondés sur Gorgias pour les utilisateurs finaux dans le cadre des systèmes de trading, ou de diagnostic médical par exemple ne font pas, eux, partie de la consultation.

Une API permet d'envoyer directement des requêtes à Gorgias et de recevoir les résultats de celles-ci. Cette API est associée à un portail de documentation.

Pour découvrir la solution, les candidats peuvent consulter les vidéos suivantes :

https://youtu.be/QZVns7Q8BY8

https://youtu.be/wAvdYI-ImJs

Ils peuvent également créer un compte sur la plateforme de démonstration, accessible à cette adresse :

http://147.27.6.99:8080/

La plateforme sera mise à disposition des utilisateurs finaux :

- Au sein d'applications tierces, intégrées en API,
- Via l'interface de dashboard pour que les utilisateurs finaux puissent exécuter les systèmes et consulter les résultats. Ces interfaces pourront être développées sur la base de modèles clés en main, dans des domaines pour lesquels l'équipe AT dispose d'ores et déjà de modèles, de scenario, etc., ou bien en mode custom, en créer ex nihilo des systèmes et requêtes pour des cas d'usage particuliers.

Les utilisateurs qualifiés pourront exploiter Gorgias de deux façons :

- En interrogeant l'API (il s'agira ici d'une API demandant un certain niveau de connaissance en informatique)
- En exploitant l'interface d'authoring. On aura ici deux grands groupes d'utilisateurs :
  - Des développeurs, informaticiens, capables d'utiliser l'interface pour construire leurs propres systèmes avec l'assistance d'AT (si besoin),
  - Les équipes Argument Theory qui utiliseront l'interface pour construire des systèmes à destination de leurs clients.

Un module de paiement sera également intégré à la nouvelle plateforme. Techniquement il fera partie intégrante de la plateforme SaaS. Il sera consultable par les utilisateurs finaux depuis le site corporate Gorgias qui présentera les différentes options d'abonnement, leurs coordonnées de facturations et l'état actuel de leur formule d'accès en cours. Il est demandé aux candidats de chiffrer :

- Pour le lot UX/UI, la conception des écrans de paiement et de consultation de son compte.
- Pour le lot technique, le développement du module de paiement ainsi qu'un module associé de comptage des requêtes pour contrôle de l'usage. Le modèle précis d'abonnement n'est pas encore arrêté, tout comme le périmètre exact du comptage et des éventuels paliers de facturation ne sont pas arrêtés. Toutefois, il importe que le futur système soit en mesure de présenter un étant complet de la consommation de ressources associés à chaque compte utilisateurs (instances, temps de CPU, espace disque, bande passante, etc.)

Plus précisément, voici comment fonctionne le développement de la connaissance utilisée par Gorgias dans son raisonnement pour lesquels de l'UX/UI est attendue :

3. LOT 1. UX / UI

Il sera demandé au prestataire de respecter les règles de l'art en matière d'UX design. Le candidat devra démontrer sa capacité à réaliser des interfaces innovantes, au service de l'utilisateur final, non seulement en respectant les règles de l'art mais également en fournissant des références concrètes dans la création ou la refonte d'interfaces au-delà des sites web.

L'équipe projet jugera les compétences des candidats aux regards des grands principes de l'UX design. Nous donnons ici quelques axes généraux pour guider la réponse des candidats sans pour autant être exhaustifs :

> 1. L'affordance des composants
> 2. Le feedback à l'utilisateur
> 3. Les limites des actions de l'utilisateur
> 4. La cohérence et les standards
> 5. Le choix des mots
> 6. La fluidité des actions de l'utilisateur
> 7. Fournir des sorties de secours à l'utilisateur

Nous détaillons ici les principaux personae et cas d'usage de l'application, attendant des candidats qu'ils proposent l'UX relative à chacun des parcours ainsi que l'UI, déclinée de l'identité Argument Theory.

Les candidats devront justifier, dans leur offre, d'une expertise tant méthodologie que pratique dans la création d'expériences de navigation et d'usage efficaces dans le cadre d'application complexes.

3.1 Personae

Trois profils d'utilisateurs sont identifiés :

> Equipe Argument Theory,
> Développeur tiers
> Utilisateur final

Les utilisateurs Equipe Argument Theory et Développeur tiers utilisent l'interface d'authoring ainsi que l'interface dite Dashboard. Les utilisateurs finaux sont donc, les développeurs qui consultent le résultat de leurs travaux mais également des utilisateurs qui, eux, ne consultent que les résultats sans être en mesure de

développer des scenarios.

Les développeurs utilisant l'API pour requêter l'application et les utilisateurs finaux consultant les résultats ne sont pas ici concernés dans la mesure où l'UX/UI qu'ils exploitent ne relève pas d'Argument Theory.

De même, les dashboards ne sont pas compris dans cette consultation.

3.2 Parcours

Les parcours utilisateurs décrits ici sont volontairement simplifiés pour laisser aux candidats la liberté d'exprimer leurs ambitions et leur vision du système d'authoring. Il est demandé aux candidats de présenter dans leur offre des parcours cohérents de bout en bout. S'ils ont des questions quant à l'utilisation de l'interface d'authoring, l'équipe AT et Erganeo est disponible pour répondre à leurs questions.

Parcours 1 : utilisateur AT ou développeur créant son propre système

> L'utilisateur accède à la plateforme et s'identifie
> Il crée un projet pour lequel il souhaite créer un nouveau système
> Il accède à l'écran d'authoring et démarre la création d'un nouveau système
> L'utilisateur passe chacune des étapes en utilisant les écrans de configuration
> L'utilisateur sauvegarde son système
> L'utilisateur utilise le système et lance des requêtes pour le tester.

Parcours 2 : Développeur venant éditer un système existant

> L'utilisateur accède à la plateforme et s'identifie
> Il accède à son tableau de bord présentant la liste des systèmes auxquels il a accès
> Il accède à un système en particulier et met à jour la connaissance du système (ajout d'options, de croyances, de préférences, ...).
> Il exécute le système et consulte le résultat

Parcours 3 : Instanciation par l'équipe AT d'un système générique pour un client particulier

> L'utilisateur accès la plateforme et s'identifie
> Il accède à son tableau de bord et sélectionne dans une liste le système générique qu'il souhaite instancier.
> Il donne une identité au système (code unique d'identification du client)
> Il personnalise / adapte la connaissance du système selon les

exigences du client

> Il exécute le système et consulte le résultat

4. LOT 2. DÉVELOPPEMENT TECHNIQUE

Dans ce lot il est demandé aux candidats de construire et de soumettre une offre pour le développement technique de l'environnement mettant à disposition et hébergeant l'applicatif Gorgias en mode SaaS.

Schéma 3 : schéma général de fonctionnement technique de Gorgias

Les candidats sont libres de proposer des évolutions de cette architecture, voire de proposer leurs propres solutions sur la base des services demandés.

4.1 General Description

This entire structure represented in the above diagram constitutes the "Software as a Service" part of Argument Theory. In case a customer requires having the engine internally, it is enough to deploy a sub-architecture (load balancer and set of Docker containers).

4.2 The main web server/router

Is in charge of:

> dealing with client authentication (through access to the client database)

> Creation of container according to customer configuration information (Retrieved from the CLIENT database, itself a Docker image). ***One set of identical containers per login below a load balancer***. This is to deal with requests load from some customers. The load balancer will balance the load amongst Docker container. Therefor Docker containers below the load balancer should be created dynamically according to the load.

> For premium customers we may need to route client communication to the dedicated load balancer (in and out). Since one set of containers is dedicated to a customer, we need to make use at this level of sticky sessions.

4.3 Client Docker Image (instantiated as multiple containers for load balancing)

Each image contains a web server with a REST API that communicates with a native Java API, which in turns dialogs with the PROLOG engine using Gorgias and the theory. Here we need to explain the way it will work:

- Upon customer login (at the main Web Server level), a set of Docker containers (from the Docker image) are instantiated with the relevant Theory.
- The client will be asserting background knowledge (fully instantiated facts and beliefs) to all these containers at the same time.
- The client will then send queries to the system (queries will be load balanced to the most available container, but since they will all contain the same knowledge, it will return an accurate answer).
- Once all the queries relating to this specific background knowledge have been performed, the background knowledge must be flushed.
- And so on…

Below a small specification of the APIs:

4.4 Java API (indicative – subject to review)

Void init(Id clientId): this will init the containers for a given client. It will retrieve from the database according to the client id all the relevant information:

> Theory location (that will be automatically loaded)
> Vocabulary (in case a translation must be made between the background knowledge pairs (name, value) received from the client and the way it is encoded in the Theory. In the form (input_name, predicate_name, arity) (arity, integer specifying the amount of arguments).

Id OpenSession() : once called a new session cannot be opened until CloseSession is called.
During a session background knowledge can be asserted to the Prolog engine. Then Gorgias can be queried as many times as necessary. But knowledge asserted cannot be changed.
For that CloseSession() must be called in order to retract all information asserted. Then a new session with new knowledge can be started.

void CloseSession(Id sessionId) : retract all asserted information (flush all background knowledge).

boolean ReloadTheory(String filename) : re-loads a theory (making sure the previous one is unloaded with unload_file JPL predicate). This is for dynamic

change of theory. This is for future use (in case theories can be edited online and reloaded dynamically).

AssertKnowledge(String[] elements, String[] values, Boolean translation): assert(element(value)) for all pairs (elements, value). Used to assert background knowledge before making a query to Gorgias. It might need a translation according to the vocabulary loaded during initialization.

This method will assert to all available containers below the load balancer. WARNING: in this case CloseSession will need to use the "abolish" predicate in order to flush the knowledge.

AssertKnowledge(String[] elements, String[] values): Creates a knowledge file before loading this file to the Prolog engine. Used to assert background knowledge before making a query to Gorgias.

This method will assert to all available containers below the load balancer. WARNING: in this case CloseSession will need to use the "unload_file" predicate in order to flush the knowledge.

List<Map<String, Term>> Query (string query_name, string[] params, string[] values, int maxAnswers, int maxMilliseconds) :

Will send a query to Gorgias in the form:

prove([query_name(param1(value1), param2(value2),.....)], EXPL)

Returns null if the result is false. Else returns the explanation in the form of a List of Maps. Each element of the list is an answer (Map of tuples <variable name, value> representing the answer). The List will not contain more than **maxAnswers** entries (therefore if only one answer is needed, there is no need to query all the possible answers). If **maxAnswers** is set to NOMAX (say -1), then returns all the possible answers. Similarly, **maxMilliseconds** can define a maximum time to wait for answers. The system returns the answers found within the time limit set. It can be cumulative with **maxAnswers.** Used to query Gorgias for a particular grounded argumentative theory that corresponds to the background knowledge of a specific instance of the problem to solve (e.g. evaluate a risk, decide about a mitigation factor). Therefore AssertKnowledge must be called before. Explanation could also provide which is the trace of the reasoning (i.e. all the grounded instances of the different (levels of) rules that have been triggered for proving the query).

This call will be directed to the most available container below the load balancer.

String[][] RequestHistory(Id sessionId) : does not make call to Gorgias, but directly to the database (audit trail). For a given session id, will return all database rows with the given *sessionId*. This is used for historical search (audit trail).

This call will be directed to the most available container below the load balancer.

String[][] RequestHistory(Date from, Date to): same as above, but returns all activity in a given date frame (several records).

This call will be directed to the most available container below the load balancer.

So the java api is the low level communication channel to Gorgias and the audit trail database. It enables to assert knowledge to Gorgias, query Gorgias and make calls to the audit trail database. It needs to be organized in sessions to deal with the change in the environment between two queries done to Gorgias.

REST API (makes direct calls to the java api):

Put: openSession: direct call to java api (modification of system status)
Put: closeSession: direct call to java api (modification of system status)
Put: reloadTheory: direct call to java api (modification of system status)
Post: postKnowledge : json set of information in the request body. Makes direct call to **AssertKnowledge(String[] elements, String[] values)** of the java api. Addition of information.
Get: Query : direct call to java api (retrieval of information). Returns json format of the list. JSON really adapted for this.
Get: RequestHistory: @requestparam, returns json in respond body. Call to the java api to retrieve information directly from the audit trail database.

4.5 Audit Trail Database Structure

Storing all api calls to the database is crucial in order to be able to retroactively justify decisions made by Gorgias to the customer.

This database will be the main data source for creating reports to the customer.

This database will be the main data source for the billing (on a per request basis).

There are several ways to design the database, but the simplest would be to have one table with 4 fields:

(sessionId, date-time, action, outcomes)

Outcome would be the query result in the most complex case (all other cases, outcome is not really relevant)=> a json string

Therefore:

OpenSession will create an entry with for example (34655hjds4656, 28/04/2019 12:01:45 cet, session opened, empty)

CloseSession will create an entry with for example (34655hjds4656, 28/04/2019 12:12:03 cet, session closed, empty)

Query will create an entry with for example (34655hjds4656, 28/04/2019 12:12:03 cet, query_name(a=45, b=56), false)

.....

4.6 Customer Database

All information that is needed to identify the customer and detail its configuration.

4.7 Payment

A billing module had to be implemented in the new platform.

This billing module needs to have the following functionalities:

> Allow payment with the main credit cards on the market
> Compatibility with the main browsers
> Client account management (personal data, payment history, status, ...)
> Bills generation
> Data consumption monitoring (CPU, bandwidth, disk space, ...)

Please see also, appendix 1 Gorgias API Doc PDF file fur forther information about Gorgias API.

5. LIVRABLES ATTENDUS

5.1 Lot 1 – UX/UI

5.1.1 Liste des livrables attendus

Il est attendu des candidats qu'ils fournissent dans leur réponse les éléments suivants :

> Références sur des projets similaires (création d'expériences utilisateurs dans des environnements digitaux innovants)
> Équipe projet dédiée
> UX : Wireframe High Fidelity des différents parcours décrits dans le présenta cahier des charges ainsi que des écrans de paiement et de gestion de son compte utilisateur
> UI : Maquette de l'un de ces parcours.
> Liste complète des templates prévus pour livraison. Il est attendu des livraisons au format CSS / HTML pour intégration dans le futur site de la plateforme par le prestataire en charge du développement technique de celle-ci.

> Organisation et gouvernance
> Mode de collaboration et de discussion autour des travaux tout au long de la vie du projet
> Planning prévisionnel
> Budget
> Engagement quant à l'utilisation privilégiée d'images libres de droits. Si des achats sont à prévoir, il est demandé aux candidats de privilégier Adobe Photostock.
> Engagement quant à la transmission de toutes les sources associées aux livrables projet (fichiers sources, iconographie, pictogrammes, ...)

5.1.2 Critères de sélection

Les critères de sélection des offres sont les suivants :

> Expérience en design / fonctionnel / SaaS
> Une première expérience dans le développement d'interfaces dans le monde de l'IA ou plateformes de services en ligne sera un plus.
> Qualité des propositions UX et originalité des propositions faites par les candidats pour enrichir l'expérience utilisateur
> Qualité de la mise en œuvre de l'identité de marque Argument Theory au sein de la plateforme développée
> Respect du planning et de l'enveloppe budgétaire
> Adéquation de l'interface avec les besoins de l'utilisateur (en fonction des profils et des différents use cases + intégration de différentes instances Gorgias)
> Expérience sur design partie collaboration (par exemple pour création d'une requête en plusieurs personnes) et logique marketplace (pour partage des différentes "bibliothèques" d'arguments et d'options déjà créés

Les modalités de réponse au présent cahier des charges englobent notamment :

> la possibilité de se positionner sur un ou plusieurs lots
> la possibilité de proposer un lot avec un partenaire, merci de nous le préciser le cas échéant et de ne pas diffuser le présent cahier des charges avant qu'un NDA ne nous ait été retourné signé de leur part (cf mention 2.1.6 Propriété intellectuelle)

5.2 Lot 2 – Réalisation de la plateforme SaaS

Il est demandé aux candidats de :

- Chiffrer la réalisation de l'application web d'accès à Gorgias,
- Chiffrer le développement de tous les services nécessaires pour l'application
- Chiffrer le développement du module de paiement et présenter le coût build / run d'une telle solution (abonnement aux solutions tierces utilisées pour le

paiement).

- Chiffrer l'hébergement de l'application web ainsi que Gorgias.
- Fournir une offre de tierce maintenance applicative présentant clairement les TJM appliqués pour les développements complémentaires ainsi que les GTI et GTR associées à l'offre d'hébergement
- Proposer un PRA en option
- Fournir des références concrètes et récentes de réalisation de plateformes SaaS sur des technologies et solutions équivalentes.
- Fournir des références concrètes et récentes dans l'hébergement et la maintenance de plateformes en mode SaaS.
- Fournir un organigramme projet clair et précis quant aux ressources disponibles et leur situation (ressources internes ou externes à la société) ainsi qu'un CV détaillé des membres de l'équipe de pilotage et de développement.
- Fournir un planning précis de livraison des différents modules applicatifs.
- Fournir un tableau des risques projet.
- Fournir une solution d'hébergement située de préférence en France, sinon en l'Union Européenne.