

# ÉPREUVE SYNTHÈSE - VOLET A

## PIZZA PLANET

### MISE EN SITUATION

Lors de vos récentes explorations, vous avez découvert dans un coin reculé de la galaxie une civilisation de monstre se nourrissant uniquement de pizza. Vous leur avez proposé d'ouvrir des pizzerias à travers la galaxie pour offrir des pizzas à l'ensemble de la population. Pour faciliter les négociations, vous vous êtes offert pour développer les services d'échanges de données permettant de traiter les commandes aux différentes pizzerias.

### MANDAT ET REMISE

Vous devez développer l'application serveur décrite dans le document, ces routes permettront de sauvegarder et de retrouver les données du système de prise de commande des pizzerias Pizza Planet.

Vous devez remettre un projet node.js complet qui inclut uniquement les routes demandées dans ce travail.

Votre travail **devra être remis avant 23 h 59, le 17 décembre 2021** sur Léa dans une archive **zip**. Votre archive devra contenir :

- Un fichier **README.md** contenant les prénoms, noms, matricules et la lettre de chacun des membres de l'équipe ;
- Un fichier **package.json** avec les informations à propos de votre projet ;
- Tous les fichiers de code source, nécessaire à l'exécution de votre serveur (incluant server.js, .env) ;
- Un fichier **tests.json**, exporté de Postman, permettant de tester vos différentes routes ;
- Les **node\_modules** ne doivent pas être remis.

### RESSOURCES ET MODÈLE

#### PIZZERIA

```
{
  "planet": "Verizuno",
  "coord": {
    "lat": -866.523,
    "lon": -552.709
  },
  "chef": {
    "name": "Krop",
    "ancestor": "Thumera",
    "speciality": "Hot Banana Pepper"
  }
}
```

Propriété	Validation
planet	Requis Présent dans PLANETS_NAMES
coord <ul style="list-style-type: none"><li>lat</li><li>lon</li></ul>	Requis, nombre réel compris entre -1000 et 1000 Requis, nombre réel compris entre -1000 et 1000
chef <ul style="list-style-type: none"><li>name</li><li>ancestor</li><li>speciality</li></ul>	Requis Requis et présent dans MONSTER_ANCESTORS Requis et présente dans PIZZA_TOPPINGS

## CUSTOMER

```
{
  "name": "Roger Bourassa",
  "email": "RogerBourassa@gustr.com",
  "planet": "Porornania",
  "coord": {
    "lat": 434.053,
    "lon": 654.959
  },
  "phone": "5EF3640075270A44",
  "birthday": "2002-08-03",
  "referralCode": "va0eWooyae",
}
```

Propriété	Validation
name	Requis
email	Requis et unique
planet	Requis Présent dans PLANETS_NAMES
coord <ul style="list-style-type: none"> <li>lat</li> <li>lon</li> </ul>	Requis, nombre réel compris entre -1000 et 1000 Requis, nombre réel compris entre -1000 et 1000
phone	Requis et 16 caractères hexadécimaux
birthday	Requis
referralCode	Aucune validation nécessaire

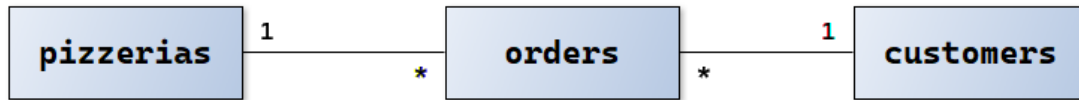
## ORDER

```
{
  "customer": "...",
  "pizzeria": "...",
  "orderDate": "2019-12-23T10:12:18.695Z",
  "pizzas": [
    {
      "toppings": ["Sausage", "Apple", "Grapes"],
      "size": "Large",
      "price": 306.578
    },
    {
      "toppings": ["Garlic", "Kiwi", "Pineapple", "Salami", "Bacon"],
      "size": "X-Large",
      "price": 712.088
    },
    {
      "toppings": ["Beef", "Spinach", "Anchovies", "Coconut"],
      "size": "Large",
      "price": 328.536
    }
  ]
}
```

Propriété	Validation
pizzeria	Requis Référence vers un ObjectId d'une Pizzeria
customer	Requis Référence vers un ObjectId d'un Customer
orderDate	Requis Date avec valeur par défaut du moment présent
pizzas <ul style="list-style-type: none"> <li>size</li> <li>price</li> <li>topping</li> </ul>	Collection de pizza Requis et présent dans PIZZA_SIZES Requis et nombre réel Collection de chaînes de caractères présentent dans PIZZA_TOPPINGS

## BASE DE DONNÉES

Votre base de données doit être composée de 3 collections présentées dans le diagramme suivant :



Vous pouvez ajouter les collections dans votre base de données du TP01 (<https://cloud.mongodb.com/>), si vous le souhaitez ou de créer une nouvelle base de données. N'oubliez de rendre disponible votre base de données pour toutes les adresses IP.

Une fois vos collections créées, vous devez ajouter les données des fichiers JSON fournis dans chacune des collections.

## SERVICES D'ÉCHANGE DE DONNÉES

Chaque membre de l'équipe doit développer les 3 routes lui étant attribuées dans la prochaine section.

### PIZZERIA

P1 - Obtenir toutes les pizzerias		Équipier B
Méthode HTTP	GET	
URL	/pizzerias	
Paramètres d'URL	page : La page demandée limit : Le maximum de document d'une page speciality :Retourner seulement les pizzerias ayant un chef dont la spécialité est celle fournie	
Corps de requête	Aucun	
Succès	200 – OK La représentation JSON transformée d'une collection paginée des pizzerias demandées	
Échec	500 – Internal Server Error	
Informations supplémentaires	Le nombre de documents par page par défaut doit être de <b>25</b> Le nombre de documents maximum par page doit être de <b>50</b> La pagination doit être trié en ordre croissant de nom de chef (chef.name) Lorsque le paramètre d'URL speciality est présent n'oublier pas d'ajuster le code pour compter le nombre de documents. Un test avec la speciality Mango pourrait être intéressant.	

P2 - Obtenir une pizzeria spécifique		Équipier A
Méthode HTTP	GET	
URL	/pizzerias/:idPizzeria	
Paramètres d'URL	embed=orders : Permet d'ajouter la représentation complète de la collection des commandes de la pizzeria portant le idPizzeria	
Corps de requête	Aucun	
Succès	200 – OK La représentation JSON transformée de la pizzeria portant le idPizzeria	
Échec	404 – Not Found Lorsque la pizzeria demandée n'existe pas 500 – Internal Server Error	
Informations supplémentaires	Aucune	

P3 - Ajouter une pizzeria		Équipier C
<b>Méthode HTTP</b>	<b>POST</b>	
<b>URL</b>	/pizzerias	
<b>Paramètres d'URL</b>	<b>_body</b> : Si faux ( <b>false</b> ), aucun corps de réponse n'est retourné	
<b>Corps de requête</b>	La représentation JSON de la pizzeria	
<b>Succès</b>	<b>201 – Created</b> La représentation JSON transformée de la pizzeria ajoutée En-tête de réponse <b>Location</b> contenant le <b>href</b> de la pizzeria ajoutée <b>204 – No Content</b> Lorsque le paramètre d'URL <b>_body</b> est à faux En-tête de réponse <b>Location</b> contenant le <b>href</b> de la pizzeria ajoutée	
<b>Échec</b>	<b>422 – Unprocessable Entity</b> Lors d'une erreur de validation de données <b>500 – Internal Server Error</b>	
<b>Informations supplémentaires</b>	Vous devez validez les données du corps de la requête avec la technique illustrée en classe avant de faire l'insertion dans la base de données.	

## CUSTOMER

C1 - Ajouter un client		Équipier B
<b>Méthode HTTP</b>	<b>POST</b>	
<b>URL</b>	/customers	
<b>Paramètres d'URL</b>	<b>_body</b> : Si faux ( <b>false</b> ), aucun corps de réponse n'est retourné	
<b>Corps de requête</b>	La représentation JSON du client à ajouter	
<b>Succès</b>	<b>200 – OK</b> La représentation JSON transformée du client ajouté En-tête de réponse <b>Location</b> contenant le <b>href</b> du client ajouté <b>204 – No Content</b> Lorsque le paramètre d'URL <b>_body</b> est à faux En-tête de réponse <b>Location</b> contenant le <b>href</b> du client ajouté	
<b>Échec</b>	<b>409 – Conflit</b> Lorsque le courriel du client est déjà existant dans la base de données <b>422 – Unprocessable Entity</b> Lors d'une erreur de validation de données <b>500 – Internal Server Error</b>	
<b>Informations supplémentaires</b>	Vous devez validez les données du corps de la requête avec la technique illustrée en classe avant de faire l'insertion dans la base de données.	

C2 - Mise à jour complète d'un client		Équipier A
Méthode HTTP	PUT	
URL	/customers/:idCustomer	
Paramètres d'URL	_body : Si faux (false), aucun corps de réponse n'est retourné	
Corps de requête	Une représentation JSON complète des modifications à apporter au client	
Succès	201 – Created La représentation JSON transformée du client venant d'être modifié 204 – No Content Lorsque le paramètre d'URL _body est à faux	
Échec	404 – Not Found Lorsque le client à mettre à jour n'existe pas 409 – Conflict Lorsque le courriel du client est déjà existant dans la base de données 422 – Unprocessable Entity Lors d'une erreur de validation de données 500 – Internal Server Error	
Informations supplémentaires	Vous devez valider les données du corps de la requête avec la technique illustrée en classe avant de faire l'insertion dans la base de données.	

C3 - Obtenir tous les clients		Équipier A
Méthode HTTP	/GET	
URL	/customers	
Paramètres d'URL	page : La page demandée limit : Le maximum de document d'une page planet :Retourner seulement les clients situés sur la planète possédant le nom fourni	
Corps de requête	Aucun	
Succès	200 – OK La représentation JSON transformée d'une collection paginée des clients demandés	
Échec	500 – Internal Server Error	
Informations supplémentaires	Le nombre de documents par page par défaut doit être de <b>20</b> Le nombre de documents maximum par page doit être de <b>40</b> La pagination doit être trié en ordre croissant de date de naissance (birthday) Lorsque le paramètre d'URL planet est présent n'oublier pas d'ajuster le code pour compter le nombre de document. Un test avec la planet Pualia pourrait être intéressant.	

C4 - Obtenir un client spécifique		Équipier C
Méthode HTTP	GET	
URL	/customers/:idCustomer	
Paramètres d'URL	embed=orders : Permet d'ajouter la représentation complète de la collection des commandes réalisées par le client portant le idCustomer	
Corps de requête	Aucun	
Succès	200 – OK La représentation JSON transformée du client spécifique portant le idCustomer	
Échec	404 – Not Found Lorsque le client demandé n'existe pas 500 – Internal Server Error	
Informations supplémentaires	Aucune	

## ORDER

O1 - Obtenir toutes les commandes		Équipier C
Méthode HTTP	GET	
URL	/orders	
Paramètres d'URL	page : La page demandée limit : Le maximum de document d'une page topping :Retourner seulement les commandes comportant au moins une pizza garnie de la garniture fournie	
Corps de requête	Aucun	
Succès	200 – OK La représentation transformée JSON d'une collection paginée des commandes demandées	
Échec	500 – Internal Server Error	
Informations supplémentaires	Le nombre de documents par page par défaut doit être de <b>10</b> Le nombre de documents maximum par page doit être de <b>30</b> La pagination doit être trié en ordre décroissant de date de commande (orderDate) Lorsque le paramètre d'URL topping est présent n'oublier d'ajuster le code pour compter le nombre de document. Un test avec le topping Jalapeño pourrait être intéressant.	

O2 - Obtenir une commande spécifique d'une pizzeria		Équipier B
Méthode HTTP	GET	
URL	/pizzerias/:idPizzeria/orders/:idOrder	
Paramètres d'URL	embed=customer : Permet d'ajouter la représentation complète du client dans la réponse	
Corps de requête	Aucun	
Succès	200 – OK La représentation transformée JSON de la commande spécifique	
Échec	404 – Not Found La pizzeria ou la commande spécifique n'existe pas 500 – Internal Server Error	
Informations supplémentaires	La commande possédant le idOrder doit appartenir à la pizzeria avec le idPizzeria. Dans le cas contraire retourner une erreur 404 – Not Found.	

## RÉPONSES

Dans toutes les réponses vous devez transformer les ObjectId en lien href comme vu en cours de session.

```
"href": "http://localhost:7187/pizzerias/5fc125a21633312494c41e86/orders/5fc125d0df76f44ce83ceb09"
```

Si l'ObjectId fait référence à un document d'une autre collection, le lien href doit être intégré dans un objet.

```
{
  "customer": {
    "href": "http://localhost:7187/customers/5fc125a21633312494c4296e"
  },
  "pizzeria": {
    "href": "http://localhost:7187/pizzerias/5fc125a21633312494c41e86"
  }
}
```

Vos services d'échange doivent répondre avec les représentations JSON suivantes.

### PIZZERIA

```
{
  "coord": {
    "lat": 637.775,
    "lon": 752.57
  },
  "chef": {
    "name": "Hazn",
    "ancestor": "Jantump",
    "speciality": "Beef"
  },
  "planet": "Sigonides",
  "href": "http://localhost:7187/pizzerias/5fc125a21633312494c41ba5",
  "lightspeed": "[Sigonides]@(637.775;752.57)"
}
```


### TRANSFORMATION

Propriété	Transformation
<b>href</b>	Le lien href pointant vers la ressource pizzeria
<b>lightspeed</b>	Dois être de ce format : <b>[planet]@(coord.lat;coord.lon)</b>

### CUSTOMER

```
{
  "coord": {
    "lat": -476.137,
    "lon": -797.893
  },
  "name": "Esperanza Bourassa",
  "planet": "Porornania",
  "referralCode": "Mah4baPh",
  "email": "EsperanzaBourassa@cuvox.de",
  "phone": "[B97F]8D94-725048E7",
  "birthday": "1979-10-24T00:00:00.000Z",
  "href": "http://localhost:7187/customers/5fc125a21633312494c4258a",
  "age": 41,
  "lightspeed": "[Porornania]@(-476.137;-797.893)"
}
```

### TRANSFORMATION

Propriété	Transformation
<b>href</b>	Le lien href pointant vers la ressource client
<b>phone</b>	Dois être dans le format [4c]4c-6c@2c (c = nombre de caractères) Exemple : <div style="border: 1px solid black; padding: 2px; display: inline-block;">           B97F8D94725048E7            [B97F]8D94-725048@E7         </div> 
<b>age</b>	Vous devez calculer l'âge du client, la bibliothèque day.js sera utile
<b>lightspeed</b>	Dois être de ce format : <b>[planet]@(coord.lat;coord.lon)</b>

## ORDER

```
{
  "customer": {
    "href": "http://localhost:7187/customers/5fc125a21633312494c4296e"
  },
  "pizzeria": {
    "href": "http://localhost:7187/pizzerias/5fc125a21633312494c41e86"
  },
  "orderDate": "2019-11-28T16:27:52.395Z",
  "pizzas": [
    {
      "toppings": [
        "Orange",
        "Mango",
        "Ham",
        "Tuna",
        "Green Olive"
      ],
      "size": "Large",
      "price": 296.45
    },
    {
      "toppings": [
        "Jalapeño",
        "Bruschetta",
        "Beef"
      ],
      "size": "Small",
      "price": 26.898
    }
  ],
  "href": "http://localhost:7187/pizzerias/5fc125a21633312494c41e86/orders/5fc125d0df76f44ce83ceb09",
  "subTotal": 323.348,
  "taxeRates": 0.0087,
  "taxes": 2.813,
  "total": 326.161
}
```

## TRANSFORMATION

Propriété	Transformation
<b>customer</b>	Objet contenant le href du client associé à cette commande
<b>pizzeria</b>	Objet contenant le href de la pizzeria associée à cette commande
<b>subTotal</b>	Somme de l'ensemble des prix ( <b>price</b> ) des pizzas de la commande
<b>taxeRates</b>	Constante de <b>0,87%</b>
<b>taxes</b>	Le montant de taxe applicable à la commande
<b>total</b>	Le sous-total additionné au montant des taxes

Tous les nombres doivent être retournés avec 3 chiffres après la virgule.

## VALIDATION

Pour procéder aux validations des données, vous procédez de deux manières dans le schéma `mongoose` et avec la méthode de validation vue en cours de session. La bibliothèque `express-validator` sera utilisée.

Dans le fichier fourni `constants.js`, plusieurs collections vous permettront de réaliser les validations demandées.

## LIENS UTILES

- <https://express-validator.github.io/docs/>
- <https://github.com/validatorjs/validator.js#validators>
- <https://mongoosejs.com/docs/schematypes.html#string-validators>



## PAGINATION

Lorsque la réponse demandée est une collection paginée, vous devez retourner un format comme celui-ci :

```
{
  "_metadata": {
    "hasNextPage": true,
    "page": 17,
    "limit": 25,
    "totalPages": 50,
    "totalDocument": 1250
  },
  "_links": {
    "prev": "",
    "self": "",
    "next": ""
  },
  "data": []
}
```

## FORMAT DES ERREURS

Lorsque vous retournez une erreur du groupe 400 ou 500 au client, vous devez utiliser ce format :

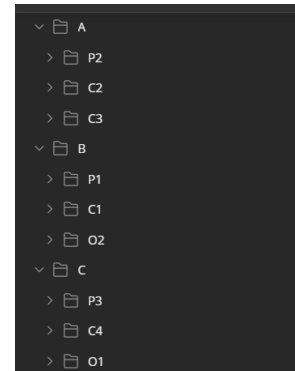
```
{
  "developerMessage": "...",
  "userMessage": "...",
  "status": 400,
  "moreInfo": "http://documentation/errors/400"
}
```

En utilisant les fichiers utilitaires fournis de manière adéquate, ceci devrait se faire automatiquement.

## TESTS

Vous devez remettre un fichier provenant d'une collection Postman, permettant de tester toutes vos routes de manière individuelle. Vous devez tester **le succès et les différentes erreurs** de chacune des routes. Chaque code de succès ou d'erreur de type **400** (400, 404, 422, etc.) de chacune des routes doit être testé. Organisez vos tests en sous-dossiers un pour chaque membre de l'équipe et un pour chacune des routes. Le fichier doit se nommer **tests.json** et être à la racine de votre projet.

**Votre serveur doit écouter sur le port 7187.**



## PONDÉRATION

Ce travail compte pour **30 %** de la note finale du cours, **15 %** seront alloués pour l'équipe et les autres **15 %** seront attribué de manière individuelle. Les points seront distribués selon les critères suivants :

<b>Critères de correction</b>	
<b>Équipier A</b>	<b>/30</b>
Fonctionnement adéquat de la route <b>P2 - Obtenir une pizzeria spécifique</b>	7
Fonctionnement adéquat de la route <b>C2 - Mise à jour complète d'un client</b>	10
Fonctionnement adéquat de la route <b>C3 - Obtenir tous les clients</b>	10
Respect du document de conception	3
<b>Équipier B</b>	<b>/30</b>
Fonctionnement adéquat de la route <b>P1 - Obtenir toutes les pizzerias</b>	10
Fonctionnement adéquat de la route <b>C1 - Ajouter un client</b>	10
Fonctionnement adéquat de la route <b>O2 - Obtenir une commande spécifique d'une pizzeria</b>	7
Respect du document de conception	3
<b>Équipier C</b>	<b>/30</b>
Fonctionnement adéquat de la route <b>P3 - Ajouter une pizzeria</b>	10
Fonctionnement adéquat de la route <b>C4 - Obtenir un client spécifique</b>	7
Fonctionnement adéquat de la route <b>O1 - Obtenir toutes les commandes d'une pizzeria</b>	10
Respect du document de conception	3
<b>Équipe</b>	<b>/30</b>
Insertion correcte des données initiales dans la base de données	3
Définitions des modèles conformes aux exigences	5
Application rigoureuse des techniques de programmation sécurisée (validation)	6
Transformations des réponses conformes aux exigences	6
Application rigoureuse des plans de tests	5
Respect du document de conception	5
<b>Total individuel</b>	<b>/30</b>
<b>Total équipe</b>	<b>/30</b>
<b>Total</b>	<b>/60</b>
<b>Total pondéré (Total ÷ 2)</b>	<b>/30</b>