[titre] L'utilisateur peut rechercher un emploi
~~Le programme sera écrit en Java~~ (techno !)

[description, 1 ou 2 phrases support à la conversation]

L'utilisateur peut voir les informations de chaque emploi qui correspond à sa recherche.

[détails, questions. Les décisions prises deviennent des tests !]

- Marco suggère d'afficher la description, le salaire et le lieu.
- Les cartes de paiement Discover sont-elles acceptées ?
- ~~On accepte les cartes Visa, Mastercard et American Express !~~ (c'est 1 test : voir au verso)

[risque technique] 2 | [coût, en story points] 4

[expectations, acceptance tests. Si tous sont ok la story est achevée]

- Tester avec une description de job vide.
- Tester avec une description de job très longue.
- Tester sans indiquer de salaire.
- Tester avec un salaire à 6 chiffres.

- Tester avec Visa, MasterCard et American Express ➜ succès !
- Tester avec Diner's Club ➜ échec !

[ une bonne story doit être **INVEST** (Bill Wake, 2003) ]

- **Independent** (*entre elles, revoir le découpage sinon*)
- **Negotiable** (*se n'est pas un contrat écrit !*)
- **Valuable** (*par l'utilisateur, l'acheteur, le client : pour le planning*)
- **Estimatable** (*par le développeur*)
- **Small** (*idéalement pas plus de qq jours, excepté les epics*)
- **Testable** (*si la story est non testable ➜ comment savoir si finie ?*)

## storypedia

acceptance tests : permet de vérifier si les stories sont achevées et développées comme attendues. Au moins écrit au dos de la story.
coût (d'une story) : estimé par les developpeurs en fonction du risque et de la compléxité. L'unité est le story point. Les US doivent faire entre 1/2 jour et 1 semaine).
epic : user story trop vaste (complexes ou composés), qui doit être découpée en stories. Utile pour matérialiser les parties futures du système : sert de support avant d'être redécoupée.
équipe d'utilisateurs (**EU**) : constituée des rôles de prioritisation, de réponse aux questions fonctionnelles des développeurs, d'utilisation du logiciel (utilisateurs finaux), de product manager, de rédaction des user stories, d'écriture et jeu des tests d'acceptance.
expectations : attentes de l'EU sous forme d'"acceptance tests".
itération : la de durée (de 1 à 4 semaines) est établie par l'équipe, puis doit rester constante durant le projet. Sert de référence pour définir la "vélocité".
planning (itération ou release) : regroupement des stories par itération selon leur priorité (coût).
release : ensemble de fonctionnalités, regroupement d'itérations.
spike : tâche / story d'exploration technique, timeboxée.
story card : partie visible de la story, mais la moins importante !
story point : unité "arbitraire" utilisée par les développeurs pour estimer la durée d'une US. Voir "coût" et "vélocité".
user story (**story**) : "Card, Conversation, Confirmation" (Jeffries).
vélocité : nombre de story points réalisés par itération. Ne sont comptabilisés que les points des stories entièrement achevées.

## planning d'itération

Prendre les stories par ordre de priorité (coût décroissant), en faisant des piles, sans dépasser la vélocité : se sont les itérations successives.
On peut "remplir" une itération avec une story de moindre importance mais plus petite.
On peut découper une story en plusieurs stories plus petites.

Une fois les itérations définies, l'EU peut regrouper des itérations successives en ensembles fonctionnels : se sont les releases.
Ex: [A:3 B:4 C1:2 F:1] [C2:2 D:4 E:4] 2 itérations, story:coût, C coupée

## tests d'acceptation

Précisent les suppositions et attentes de l'EU.
Doivent donc être rédigés en début d'itération, le plus tôt possible.
Au moins écrits au dos de la story, voire rédigés dans un framework technique permettant d'automatiser les tests.
Automatiser les tests est vital dans un développement itératif.

## processus

1. Définir la matrice utilisateur / rôle (chooseuser.com)
2. L'équipe fixe la durée d'une itération pour tout le projet (1 à 4 semaines)
3. L'équipe crée les stories sur la base d'une conversation.
4. L'EU donne les priorités aux stories en fonction de leur valeur pour l'organisation.
5. Les développeurs donnent leur avis sur les priorités (fonction du risque technique ou la complémentarité de stories)
6. Les développeurs estiment les stories en story point.
7. L'EU fait le planning d'itération.

## tips

La story doit être rédigée par l'EU, dans son langage, pour pouvoir être valorisée. Pas de story technique.
Si les développeurs n'arrivent pas estimer une story c'est probablement qu'il ne maîtrise pas le métier (discuter avec l'EU), et/ou la technique (faire un spike timeboxé), ou que la story est trop vaste (la découper).
Pour une story difficile à estimer, on pourra créer une story "spike" puis la story de réalisation qu'on estimera après que l'exploration soit faite.
On peut regrouper diverses petites tâches (debug, retouche interface graphique...) en une même story.
Une story non testable est bien souvent non fonctionnelle.
Pour plus de détails :

- 2ia.net/user-story
- "*User Stories Applied*", Mike Cohn (Addison-Wesley)
- "*Agile Estimating and Planning*", Mike Cohn.