Backlog

Wirtschaftsprojekt «px: PEAX Command Line Client»

Patrick Bucher

12.10.2019

#	User Story	Status	Story Points
1	Konfiguration sämtlicher Umgebungen	eingeplant für Sprint 1	1
2	Erweiterung der CI-Pipeline	eingeplant für Sprint 1	5
3	Login mit Zwei-Faktor-Authentifizierung	eingeplant für Sprint 1	3
4	Sichere Verwahrung der Tokens	eingeplant für Sprint 1	5
5	Handhabung mehrerer Umgebungen	eingeplant für Sprint 1	3
6	Generische `GET`-Schnittstelle	eingeplant für Sprint 1	3
	Verbesserung der Hilfe-Funktion	vorgesehen für Sprint 2	
	Automatische Aktualisierung von Tokens	vorgesehen für Sprint 2	
	Login für Agent API	vorgesehen für Sprint 2	
	Einliefern von Dokumenten per Delivey API	offen	
	Einliefern von Dokumenten per User API	offen	
	Einlieferung von Verzeichnissen mit Dokumenten	offen	
	Auflisten von Dokumenten mit Suche/Filterung	offen	
	Generische `POST`-Schnittstelle	offen	
	Generische `PUT`-Schnittstelle	offen	
	Generische `PATCH`-Schnittstelle	offen	
	Ausführung von Befehlen für mehrere Umgebungen	offen	
	Fortschrittsanzeige bei längeren Vorgängen	offen	
	Ausgabe von Tokens	offen	
	Inspektion von Tokens	offen	

Sprints

Sprint 1

Eingeplant: sechs Stories, 20 Story Points

User Stories

Die User Stories haben drei Grössen: S (small: 1 Story Point), M (medium: 3 Story Points) und L (large: 5 Story Points). Diese Grössen dienen zur relativen Einschätzung der Story-Grössen zueinander, und sollen nicht in eine Stundenplanung heruntergerechnet werden. Vielmehr sollen sie dazu dienen, übergrosse Stories als solche zu erkennen, um diese herunterbrechen zu können. Am Ende eines jeden Sprints soll eingeschätzt werden, wie viele Story Points ungefähr machbar sind.

1: Konfiguration sämtlicher Umgebungen

Als Entwickler möchte ich sämtliche relevanten PEAX-Umgebungen vorkonfiguriert haben, damit diese dem Nutzer zur Verfügung gestellt werden können.

Akzeptanzkriterien:

1. Es sollen die Umgebungen `dev`, `test`, `devpatch`, `testpatch`, `stage`, `prod`, `perf` und `prototype` zur Verfügung stehen.
2. Für jede Umgebung muss eine erreichbare URL zum jeweiligen Identity Provider und zu den relevanten APIs (User API, Admin API, Agent API) automatisch generiert werden können.

2: Erweiterung der CI-Pipeline

Als Entwickler möchte ich `px` in Skripts einbauen können, welche anschliessend in der CI-Pipeline berücksichtigt, d.h. ausgeführt werden.

Akzeptanzkriterien:

1. Der Ausführungsschritt `script` soll nur dann ausgeführt werden, wenn die vorherigen Schritte `build` und `test` erfolgreich waren.
2. Um der Pipeline ein weiteres Skript hinzufügen zu können, soll die neu erstellte Skriptdatei nur an einer einzigen Stelle (Konfigurationsdatei) hinzugefügt werden müssen.

3: Login mit Zwei-Faktor-Authentifizierung

Als Benutzer möchte ich mich per Zwei-Faktor-Authentifizierung einloggen können, damit ich `px` auch mit entsprechend konfigurierten Zugängen verwenden kann.

Akzeptanzkriterien:

1. Die Abfrage des zweiten Faktors soll interaktiv passieren.

2. Es sollen die Authentifizierungsarten SMS und OTP (One-Time Password) unterstützt werden.
3. Das Login soll auch weiterhin ohne Zwei-Faktor-Authentifizierung funktionieren.

4: Sichere Verwahrung der Tokens

Als Benutzer möchte ich, dass beim Login geholte Tokens sicher lokal verwahrt werden, damit ein Angreifer diese nicht auslesen kann.

Akzeptanzkriterien:

1. Die Credentials (Benutzername und Password) werden zu keinem Zeitpunkt lokal persistent abgespeichert.
2. Refresh Tokens, die von der Produktivumgebung (`prod`) geholt werden, dürfen standardmässig nicht im Klartext abgespeichert werden.
3. Access und Refresh Tokens von nicht-produktiven Umgebungen, sowie Access Tokens der Produktivumgebung, können lokal im Klartext abgespeichert werden, sofern dies der einfacheren Bedienbarkeit zuträglich ist (weniger Passwortabfragen durch den Keystore).
4. Der Benutzer soll das Standardverhalten für die jeweiligen Umgebungen (produktiv: nur sichere Verwahrung; nicht-produktiv: Verwahrung im Klartext) mit den Kommandozeilenparametern `-safe` bzw. `-unsafe` übersteuern können.
5. Die sichere Verwahrung der Tokens muss Windows, macOS und Linux funktionieren.
6. Auf Systemen ohne GUI soll zumindest die unsichere Variante der Token-Verwahrung funktionieren.

5: Handhabung mehrerer Umgebungen

Als Benutzer möchte ich, dass `px` meine Befehle standardmässig gegen die Umgebung ausführt, auf der ich mich zuletzt eingeloggt habe, damit ich nicht immer eine Umgebung per Kommandozeilenparameter anwählen muss.

Akzeptanzkriterien:

1. Es muss einen Unterbefehl geben, der mir die aktuelle Umgebung (d.h. die Umgebung, auf der sich der Benutzer zuletzt eingeloggt hat) anzeigt.
2. Es muss einen Unterbefehl geben, womit eine Umgebung mit bereits aktivem Logging als die Standardumgebung gesetzt werden kann.
3. Bei allen Befehlen, die gegen die API operieren, soll die Umgebung mit einem Kommandozeilenparameter `-e` bzw. `-env` spezifiziert werden können.

6: Generische `GET`-Schnittstelle

Als Benutzer möchte ich einen `get`-Befehl zur Verfügung haben, damit ich lesend auf meine Ressourcen zugreifen kann.

Akzeptanzkriterien:

1. Dem Befehl kann ein beliebiger Ressourcenpfad mitsamt Query-Parametern mitgegeben werden.
2. Die Base-URL der jeweiligen API und Umgebung wird dem Ressourcenpfad automatisch vorangestellt.
3. Im Falle eines erfolgreichen Zugriffs (`200 OK`) soll der resultierende Payload auf die Standardausgabe (`stdout`) ausgegeben werden.
4. Im Falle eines fehlerhaften Zugriffs soll der Status-Code auf die Standardfehlerausgage (`stdout`) ausgegeben werden.