

PISTL - Outil d'administration de comptes client pour la société UpClear

Walter ABELES, Daniel SIMA, Mickael TEMIM et Yukai LUO

Soutenance finale

Mardi 6 février



- **Introduction**
 - Présentation de l'entreprise
 - Rappel des objectifs du projet
 - Organisation
- **Structure du projet**
 - Architecture de l'API
 - Workflows et description des fonctionnalités
- **Tests**
 - Intégration
 - End-to-End
- **Déploiement**
- **Conclusion**
- **Kit de livraison**
- **Perspectives**
- **Démonstration**

L'entreprise UpClear et l'outil DAT

- **UpClear :**

Entreprise spécialisée dans les solutions de gestion d'entreprise automatisées, offrant une gamme de produits et de services destinés à transformer la gestion des stocks et la logistique, tel que **BluePlanner**.

- **DAT (Data Administration Tool) :**

Outil back-office d'administration de comptes client.

API Rest en Single-Page Application (SPA).

Utilisé par 4 personnes de l'entreprise.



Objectifs du projet

- Reconstruire DAT et rendre sa maintenance plus facile par les équipes en créant une réelle séparation backend et frontend.
- Reconstruction des 8 écrans existants + celui de login.
- Ajout de deux fonctionnalités supplémentaires :
 - Un écran pour poster un message sur les dashboards des clients.
 - La double authentification ou SSO.

Organisation de l'équipe

- Réunion hebdomadaire entre membres de l'équipe
- Réunion hebdomadaire avec le client
 - Tableau de bord
- Utilisation de méthodes agiles
 - Kanban : organisation des tâches entre les membres de l'équipe
 - User-Stories : organisation du développement de l'API

Nouveau DAT

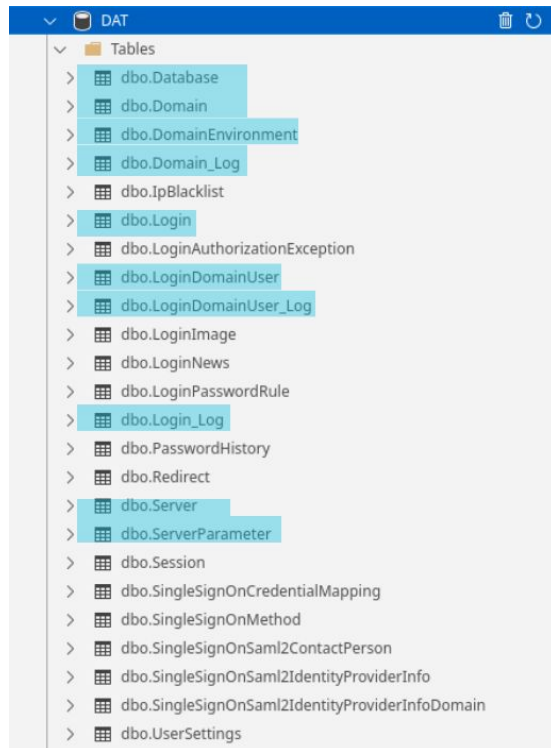
Réalisé:

- Toutes les pages de l'application (8 écrans existants + celui de login).
- La double authentification comme fonctionnalité supplémentaire.
- Déploiement avec Docker Compose.
- Kit de livraison.

Non réalisé:

- Seconde fonctionnalité supplémentaire
 - Écran pour poster un message sur les dashboards des clients.
- Déploiement sur Kubernetes

Point de départ



Tables dans la base de données DAT

- Tables fournies (database first)
 - Tables utilisées après la génération avec l'ORM Entity Framework.
- Un script.sh pour générer l'image Docker de la base de données
- Charte graphique pour uniformiser le front-end

Architecture du projet

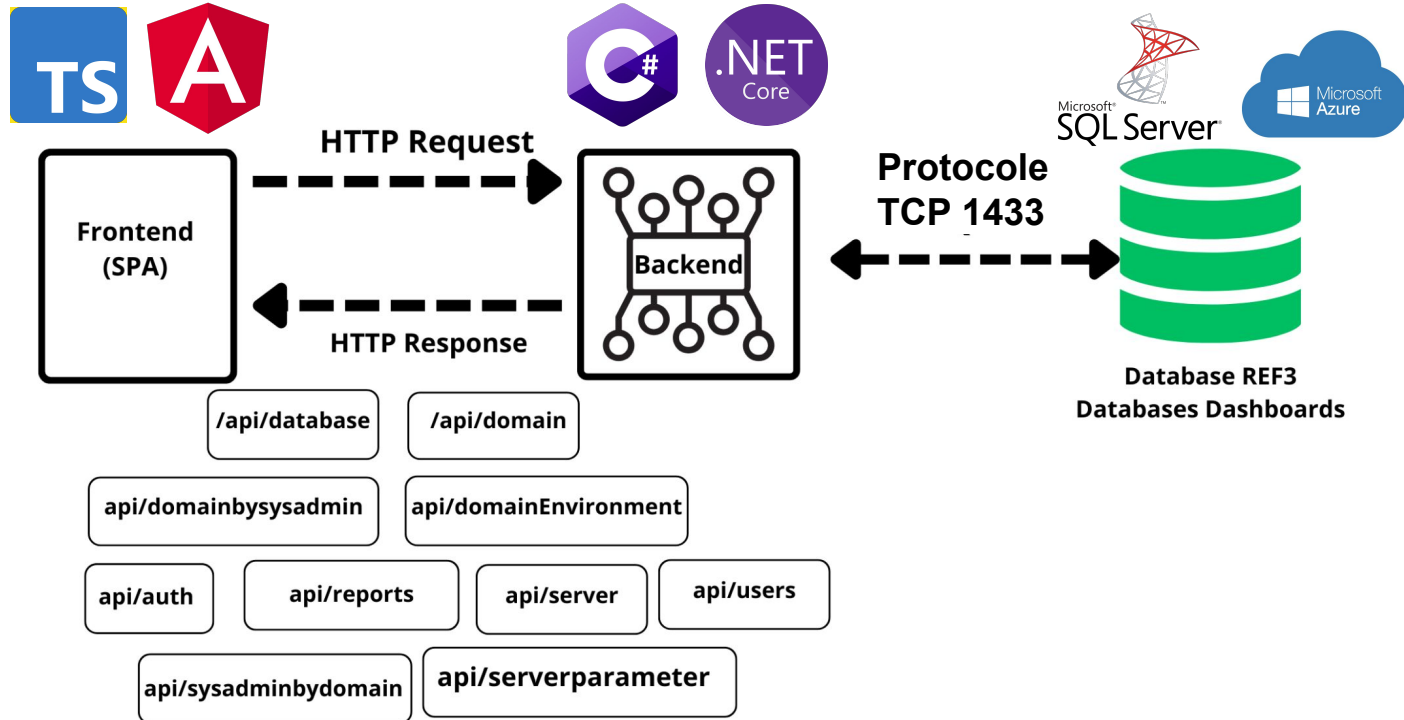
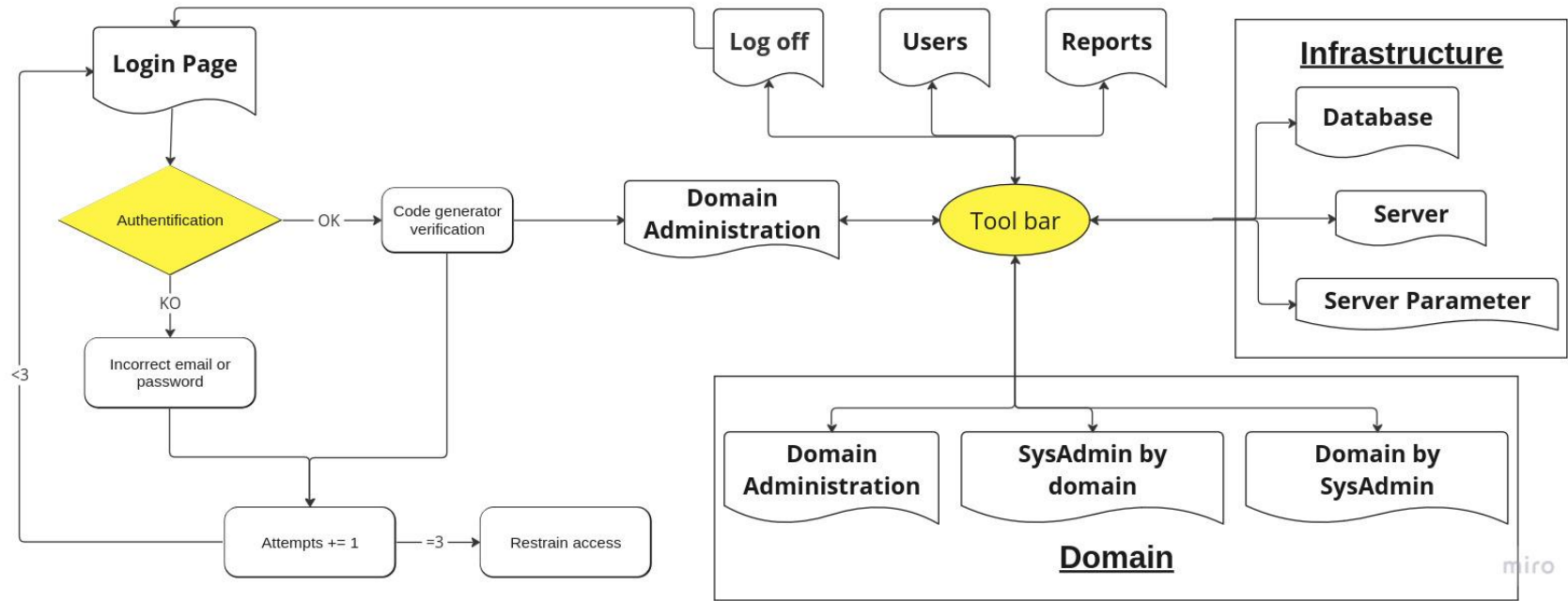



Schéma de l'architecture globale

Workflow de l'API



Workflow global de l'application

Authentification



dbo.Login

Columns
LoginID (PK, int, not null)
Email (nvarchar(200), null)
Name (nvarchar(50), not null)
Password (varbinary(max), not null)
PasswordSalt (char(32), null)
PasswordModifiedDate (datetime, null)
PasswordExpirationDate (datetime, null)
InvalidAttemptCount (int, null)
ResetPasswordEndDate (datetime, null)
ResetPasswordKey (varchar(72), null)
ResetPasswordSentCount (smallint, not null)
ResetPasswordInvalidAttemptCount (smallint, not null)
LastLoginDate (datetime, null)
TermsAccepted (bit, not null)
DATEnabled (bit, null)
Phone (varchar(50), null)
BlockedReason (nvarchar(max), null)
CreatedDate (datetime, null)
CreatedBy (nvarchar(200), null)
ModifiedDate (datetime, null)
ModifiedBy (nvarchar(200), null)

Table **Login**

- Première étape de connexion classique
 - Email + Password
- Table **Login**
 - Plusieurs champs utilisés



UpClear.
BluePlanner

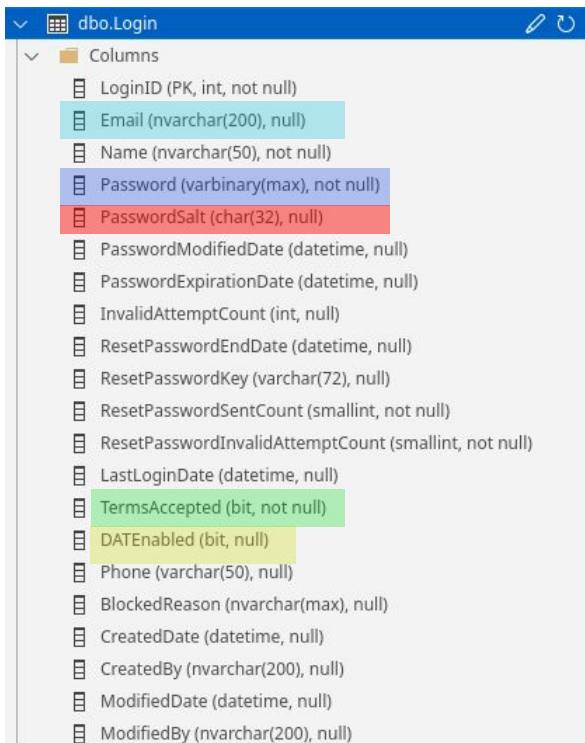
Email address

Password

Log In

Écran de login

Authentification



Columns
LoginID (PK, int, not null)
Email (nvarchar(200), null)
Name (nvarchar(50), not null)
Password (varbinary(max), not null)
PasswordSalt (char(32), null)
PasswordModifiedDate (datetime, null)
PasswordExpirationDate (datetime, null)
InvalidAttemptCount (int, null)
ResetPasswordEndDate (datetime, null)
ResetPasswordKey (varchar(72), null)
ResetPasswordSentCount (smallint, not null)
ResetPasswordInvalidAttemptCount (smallint, not null)
LastLoginDate (datetime, null)
TermsAccepted (bit, not null)
DATEnabled (bit, null)
Phone (varchar(50), null)
BlockedReason (nvarchar(max), null)
CreatedDate (datetime, null)
CreatedBy (nvarchar(200), null)
ModifiedDate (datetime, null)
ModifiedBy (nvarchar(200), null)

Table **Login**

Email - exemple: testing@upclear.com

Password - exemple: 0x59C7C86D4183A3...

- Password sous forme de chaîne de caractères + **Salt** hashé avec **SHA512**

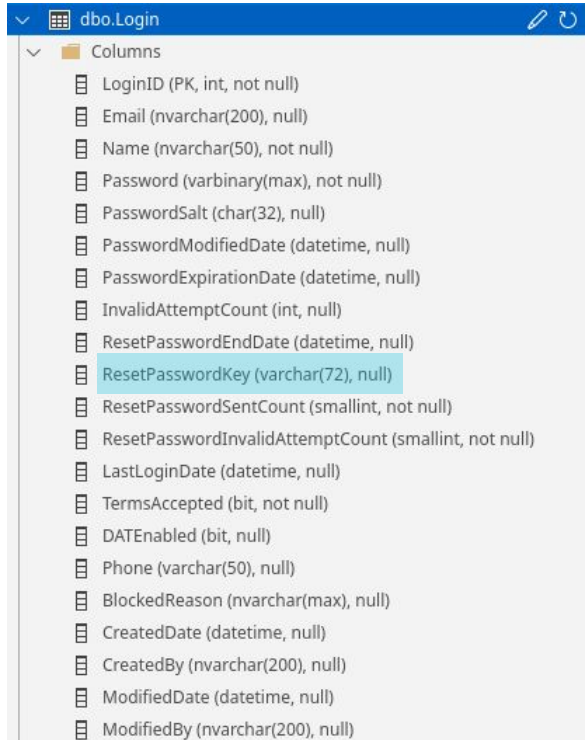
Salt - exemple: 7E4qzBmcL1WWbgfQF10d68P5oNuxDdGV

- Chaîne de caractères générée de manière aléatoire

TermsAccepted - Boolean signalant un utilisateur bloqué

DATEnabled - Boolean signalant si un utilisateur a le droit d'accès à DAT

Authentification - 2FA



Columns
LoginID (PK, int, not null)
Email (nvarchar(200), null)
Name (nvarchar(50), not null)
Password (varbinary(max), not null)
PasswordSalt (char(32), null)
PasswordModifiedDate (datetime, null)
PasswordExpirationDate (datetime, null)
InvalidAttemptCount (int, null)
ResetPasswordEndDate (datetime, null)
ResetPasswordKey (varchar(72), null)
ResetPasswordSentCount (smallint, not null)
ResetPasswordInvalidAttemptCount (smallint, not null)
LastLoginDate (datetime, null)
TermsAccepted (bit, not null)
DATEnabled (bit, null)
Phone (varchar(50), null)
BlockedReason (nvarchar(max), null)
CreatedDate (datetime, null)
CreatedBy (nvarchar(200), null)
ModifiedDate (datetime, null)
ModifiedBy (nvarchar(200), null)

Table **Login**

- Seconde étape de connexion si double authentification configurée
 - Via un generateur de codes comme Google Authenticator
 - Génération d'un **JSON Web Token** pour la session
- Table **Login**
 - **ResetPasswordKey** - Stockage du QR Code



UpClear.
BluePlanner

Code

Log In Cancel

Écran de double authentification

Domain Administration

Domain Administration

Select domain
Chanel [353] Preprod Test ProdCopy

New Edit Copy Delete

Domain Name*
Chanel

CHANEL

Edition
Light

Comment
Luxe

Parent Company
LVMH

☒ BP5

Dev Preprod Prod Test ProdCopy Staging

Database Server

Web Server
SSRS dev/tst

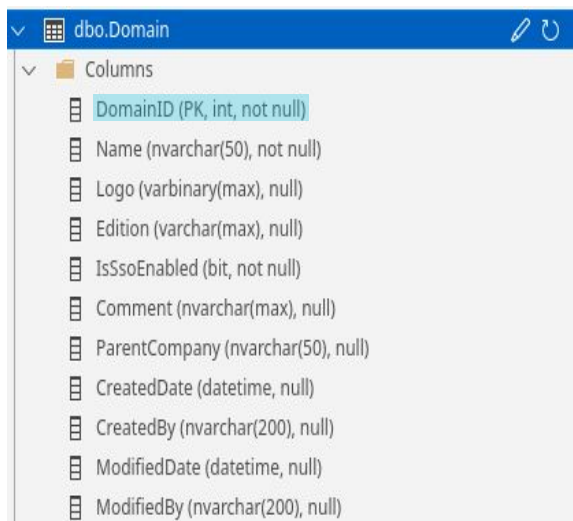
EAI FTP
SSRS dev/tst

Tableau Server
axdev Tableau Dev (UK - Azure)

SSRS Server
SSRS dev/tst

- Gestion des domaines
- Plusieurs environnements possibles par domaine
- Stockage des modification réalisés via des triggers dans la table **Domain_Log**
- Suppression d'un domain entraîne la suppression d'instances dans d'autres tables liées

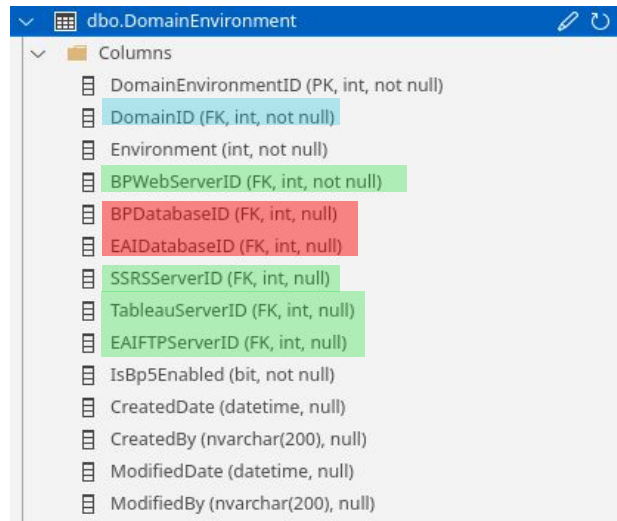
Domain Administration



dbo.Domain

Columns
DomainID (PK, int, not null)
Name (nvarchar(50), not null)
Logo (varbinary(max), null)
Edition (varchar(max), null)
IsSsoEnabled (bit, not null)
Comment (nvarchar(max), null)
ParentCompany (nvarchar(50), null)
CreatedDate (datetime, null)
CreatedBy (nvarchar(200), null)
ModifiedDate (datetime, null)
ModifiedBy (nvarchar(200), null)

Table **Domain**



dbo.DomainEnvironment

Columns
DomainEnvironmentID (PK, int, not null)
DomainID (FK, int, not null)
Environment (int, not null)
BPWebServerID (FK, int, not null)
BPDatabaseID (FK, int, null)
EADatabaseID (FK, int, null)
SSRSServerID (FK, int, null)
TableauServerID (FK, int, null)
EAIFTPServerID (FK, int, null)
IsBp5Enabled (bit, not null)
CreatedDate (datetime, null)
CreatedBy (nvarchar(200), null)
ModifiedDate (datetime, null)
ModifiedBy (nvarchar(200), null)

Table **DomainEnvironment**

Liaison domaines
-
environnements

Liaison server
-
environnements

Liaison databases
-
environnements

SysAdmin by Domain / Domain by SysAdmin

dbo.LoginDomainUser

Columns
LoginID (PK, FK, int, not null)
DomainID (PK, FK, int, not null)
UserID (PK, varchar(50), not null)
Environment (PK, int, not null)
UserName (nvarchar(200), null)
UserActive (bit, not null)
LoginEnabled (bit, not null)
LoginTypeId (int, null)
AnalyticsEnabled (bit, null)
IsLight (bit, null)
SysAdmin (bit, not null)
SysAdminStartDate (datetime, null)
SysAdminEndDate (datetime, null)
DomainLastLoginDate (datetime, null)
Comment (nvarchar(2000), null)
CreatedDate (datetime, null)
CreatedBy (nvarchar(200), null)
ModifiedDate (datetime, null)
ModifiedBy (nvarchar(200), null)

Table **LoginDomainUser**

dbo.Domain

Columns
DomainID (PK, int, not null)
Name (nvarchar(50), not null)

Table **Domain**

dbo.Login

Columns
LoginID (PK, int, not null)
Email (nvarchar(200), null)

Table **Login**

- Jointure entre la table **LoginDomainUser** et **Domain**
- Jointure entre la table **LoginDomainUser** et **Login**
- Récupération des utilisateurs grâce aux clés *DomainId* et *LoginId*

SysAdmin by Domain / Domain by SysAdmin

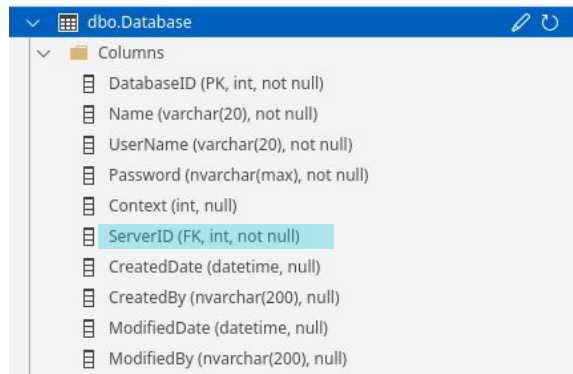
- Gérer les droits d'administrateur de chaque utilisateur pour chaque environnement dans les domaines auxquels ce dernier a accès.
- Table **LoginDomainUser** :
 - Chaque **Login** (utilisateur) peut avoir un *user* dans chaque **Domain**.
 - Chaque *user* est associé à un **Environment** de travail
 - Lorsqu'un **Login** reçoit les droits d'administration, un *user* est créé avec l'id "99999999-9999-9999-9999-999999999999"
- Chaque action réalisée sur la table **LoginDomainUser** est répertoriée dans la table **LoginDomainUser_Log** via des triggers.

Infrastructure

- Pages **Database**, **Server** et **Server Parameter**
- Gérer l'ajout, l'édition, la suppression et la copie à l'aide de méthodes POST, PUT et DELETE orchestrés par les contrôleurs sur la base de donnée.

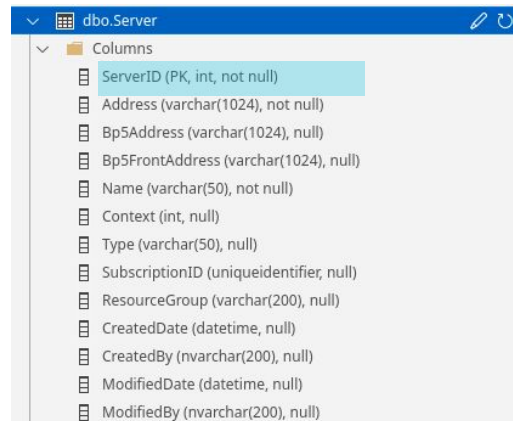
Server				
New Edit Delete				
ID	Address	Name	Context	Type
1	TestAdressUpdate	TestSeverNameUpdate	Dev	TestTypeUpdate
2	TestAddress	TestSeverName	Dev	TestType
13	https://tst1aaxgw.blueplanner.com	axdev Tableau Dev (UK - Azure)	Dev	TABLEAU
98	db4a.blueplanner.com	SSRS dev/tst	Dev	SSRS
227	172.17.81.136	dev-weu-ftp-01 (Test)	Prepro	EAI

Database et Server



dbo.Database	
Columns	
DatabaseID (PK, int, not null)	
Name (varchar(20), not null)	
UserName (varchar(20), not null)	
Password (nvarchar(max), not null)	
Context (int, null)	
ServerID (FK, int, not null)	
CreatedDate (datetime, null)	
CreatedBy (nvarchar(200), null)	
ModifiedDate (datetime, null)	
ModifiedBy (nvarchar(200), null)	

Table **Database**



dbo.Server	
Columns	
ServerID (PK, int, not null)	
Address (varchar(1024), not null)	
Bp5Address (varchar(1024), null)	
Bp5FrontAddress (varchar(1024), null)	
Name (varchar(50), not null)	
Context (int, null)	
Type (varchar(50), null)	
SubscriptionID (uniqueidentifier, null)	
ResourceGroup (varchar(200), null)	
CreatedDate (datetime, null)	
CreatedBy (nvarchar(200), null)	
ModifiedDate (datetime, null)	
ModifiedBy (nvarchar(200), null)	

Table **Server**

- Jointure entre la table **Database** et **Server**

Users

dbo.Login

Columns

LoginID (PK, int, not null)
Email (nvarchar(200), null)
Name (nvarchar(50), not null)
Password (varbinary(max), not null)
PasswordSalt (char(32), null)
PasswordModifiedDate (datetime, null)
PasswordExpirationDate (datetime, null)
InvalidAttemptCount (int, null)
ResetPasswordEndDate (datetime, null)
ResetPasswordKey (varchar(72), null)
ResetPasswordSentCount (smallint, not null)
ResetPasswordInvalidAttemptCount (smallint, not null)
LastLoginDate (datetime, null)
TermsAccepted (bit, not null)
DATEabled (bit, null)
Phone (varchar(50), null)
BlockedReason (nvarchar(max), null)
CreatedDate (datetime, null)
CreatedBy (nvarchar(200), null)
ModifiedDate (datetime, null)
ModifiedBy (nvarchar(200), null)

Utilisés

- Page de gestion des utilisateurs

Sys Admin Users

New Edit Delete Reset Password Unlock

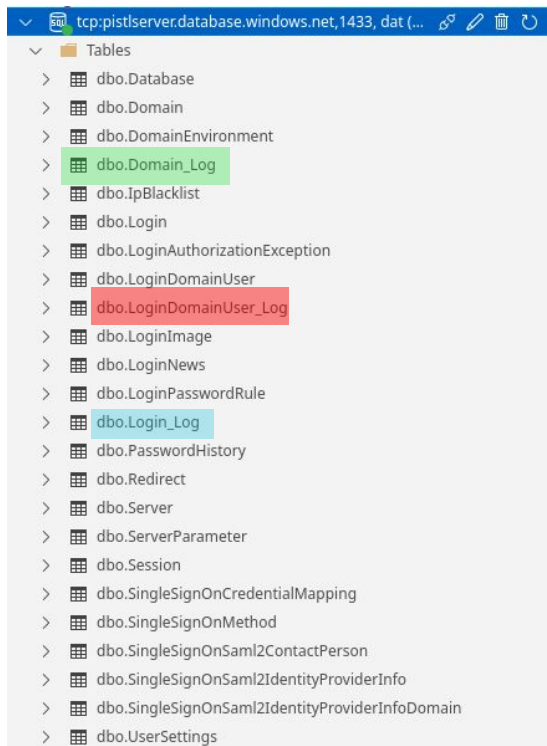
Email	DAT
local@upclear.com	✗
sso@upclear.com	✗
local.admin@upclear.com	✗
domaintemplate@upclear.com	✗
test2@example.com	✓

1 - 5 of 7

Items per page: 5

Écran page Users

Reports



- Génération des rapports sous forme **.csv** récupérant les modifications faites sur 3 tables via des trigger SQL.

- User Activity Report

- Informations de connexion

- SysAdmin by Domain History

- Informations sur les droits des SysAdmin sur les domains

- Domain Administration History

- Informations sur les domaines

Tests

- **Tests d'intégration - NUnit** (Méthode 3A)
 - Arrange
 - Act
 - Assert
- **Tests End-to-End : Cypress**
 - Vérifier le bon fonctionnement de l'outil DAT du début à la fin d'une user-story

Tests d'intégration

```
[Test]
0 références
public void TestUpdateUserAlreadyExists()
{
    // Arrange
    var controller = new SysAdminByDomainController();
    var user = new LoginDomainUserDTO
    {
        DomainId = 351,
        Environment = 4,
        LoginId = 26995,
        UserId = "99999999-9999-9999-9999-999999999999",
        ModifiedBy = "admin",
        SysAdmin = true,
        SysAdminStartDate = DateTime.Parse("2020-05-13T00:00:00"),
        SysAdminEndDate = DateTime.Parse("2021-05-13T00:00:00"),
        Comment = "dsvs",
        UserName = null
    };
    controller.UpdateUser(user);

    // Act
    var response = controller.UpdateUser(user) as OkObjectResult;

    // Assert
    Assert.Multiple(() =>
    {
        Assert.AreNotEqual(response, null);
        Assert.That(response.StatusCode, Is.EqualTo(200));

        var json = JsonConvert.SerializeObject(response.Value);
        Assert.AreNotEqual(json, null);

        Assert.IsInstanceOf<LoginDomainUserDTO>(response.Value, "Wrong type");
        var values = JsonConvert.DeserializeObject<LoginDomainUser>(json);

        // Assert that the object is not null
        Assert.AreNotEqual(values, null);
    });
}
```

Exemple d'un test sous NUnit

- Tests d'intégration réalisés pour chaque controller pour garantir la couverture du code.
- Déroulement d'un test :
 - **A**rrangement de la base de données (si besoin).
 - **A**ction du controller.
 - **A**ssertions sur les résultats obtenus.

Tests End-To-End

- **Cypress** : Outil d'automatisation de tests frontaux pour les tests de régression des applications Web
- Vérifier le bon fonctionnement de l'outil DAT du début à la fin d'une user-story.

(Voir la démonstration sur Cypress)

Déploiement - Docker

```
version: '3.8'

services:
  backend:
    build:
      context: ../backend
      dockerfile: Dockerfile
    ports:
      - "5050:5050"
    environment:
      ConnectionStrings__AzureDatabase:
        "Server=tcp:datserver2.database.windows.net,1433;
        Initial Catalog=DAT;Persist Security Info=False;
        User ID=walter;Password=Daniel123;
        MultipleActiveResultSets=False;Encrypt=True;
        TrustServerCertificate=False;
        Connection Timeout=30;"

  frontend:
    build:
      context: ../frontend
      dockerfile: Dockerfile
    ports:
      - "4200:4200"
    depends_on:
      - backend
```

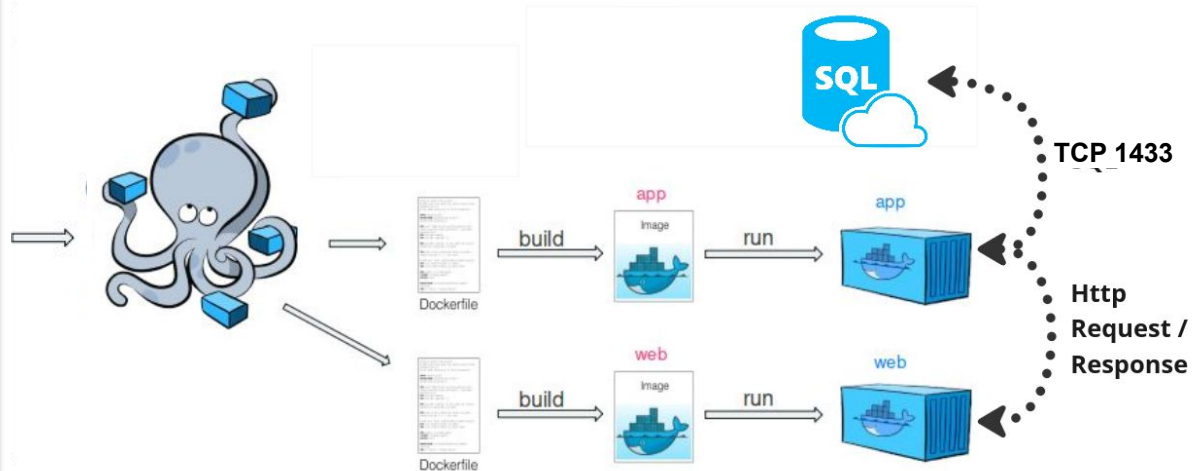
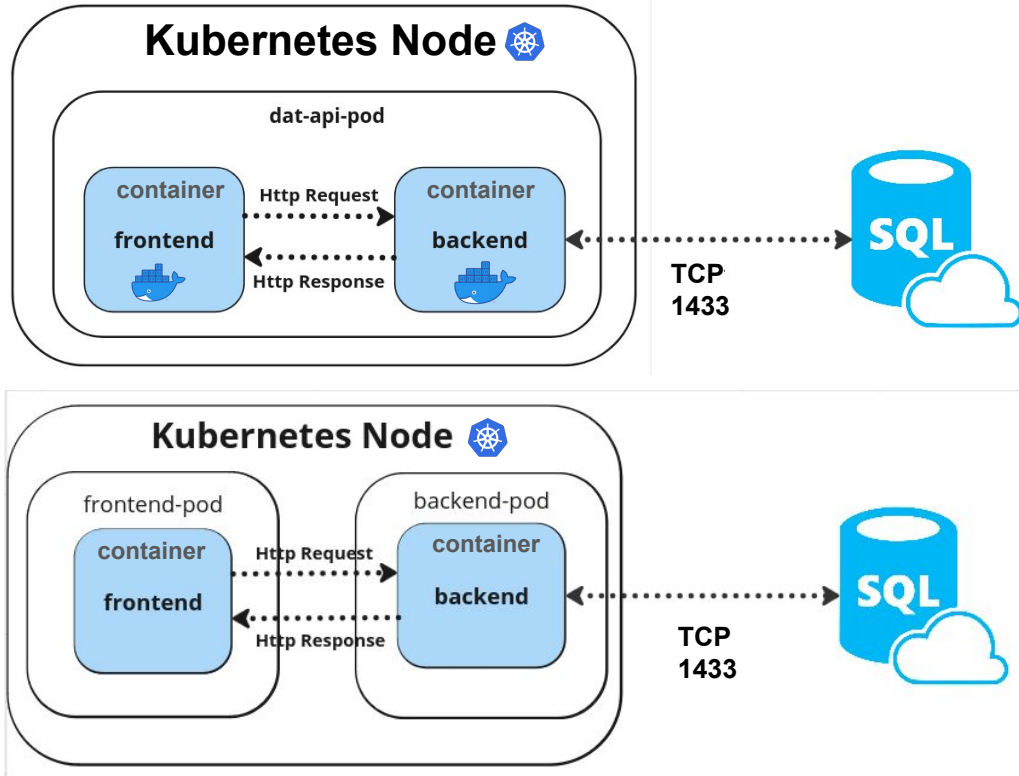


Schéma de déploiement

Déploiement - Kubernetes



- **Premier Scénario**

- 1 pod
- 1 container pour le backend
- 1 container pour le frontend

- **Second Scénario**

- 1 pod contenant le container du frontend
- 1 pod contenant le container du backend

Difficultés rencontrées

- Intégration de la BDD et de l'API dans un container Docker
 - Génération de la base à l'aide d'un script shell fournit par le client et fonctionnel après adaptation.
- Kubernetes : connexion entre le backend et la base de données Azure
 - Problèmes liés aux CORS
- Abonnement Azure expiré

Conclusion

- Projet terminé dans sa totalité sans la fonction supplémentaire de poster des messages sur les dashboards des clients.
- Déploiement d'une Azure Database générée à partir des modèles de la base fournie par le client.
- Déploiement d'un container backend et frontend dans un Docker-compose.
- Déploiement des 2 scénarios sur Kubernetes non achevés.

Kit de livraison

- Répo Github (changement de owner)
- Manuel d'utilisation et de maintenance
- Manuel de déploiement
- État final de la configuration
- Spécifications STBE et STR

Perspectives

- Traitement de la concurrence au niveau des containers
 - Interblocages au niveau de la mise à jour de la base de données.
- Liaison avec le logiciel principal, BluePlanner
- Architecture permettant l'ajout de nouvelles fonctionnalités
 - Maintenance facilitée par la documentation et la décomposition frontend / backend
- Développement de l'application en responsive

Démonstration de l'application

Merci pour votre attention !