Devoir 2

Cours LOG430

Réalisé par

Rim BEN DHAOU

Devoir 2

1. Question 1 :

1.1. Vue module :

Pour la vue module nous allons combiner 4 styles architecturaux : décomposition, utilise, généralisation et en couche.

Style architectural : décomposition

Justification : La décomposition supporte la modificabilité du système en décomposant les services du système en des modules et ces modules en de sous modules selon les préoccupations et les fonctionnalités de chaque module. Cette décomposition doit distinguer entre les modules communs utilisés par tous les types d'élection et les modules variables. Nous pouvons ainsi réutiliser des modules pour le cas des autres types d'élection. De cette façon, lors de la modification la localisation du changement sera plus claire. Cette décomposition servira en plus pour les nouveaux développeurs qui vont rejoindre l'équipe de développement. Ce style architectural facilite et accélère aussi la tâche de développement en allouant des modules séparés aux différentes équipes de développement.

Style architectural : Utilise

Justification : Ce style illustre comment les modules dépendent l'un des autres. En plus il fournit une meilleure analyse d'impact dans le cas des modifications.

Style architectural : généralisation :

Justification : La généralisation, grâce à la relation 'est un', supporte l'évolution la réutilisation et la modification du système d'un système de vote provincial vers un autre système de vote (municipal, fédéral, …). Nous pouvons présenter l'abstraction des modules pour généraliser les objets en communs et présenter les variabilités comme des modules 'fils'.

Style architectural : En couche

Justification : Nous aurons besoin de structurer le système en 3 couches

Couche présentation : pour tout ce qui est affichage des interfaces et des formulaires de connexion, de vote, …

Couche métier : traite les informations à présenter

Couche accès aux Bases de données : sauvegarde et accès aux informations des électeurs, des parties politiques, des candidats, …

L'architecture en couche permet de regrouper chaque préoccupation en une couche et rend le système plus flexible (facile à modifier et à réutiliser), moins complexe et facile à tester. Une modification dans une couche du bas niveau n'affecte pas les couches supérieures. Cette architecture assure une isolation entre les données et le logique métier et entre le logique métier et la couche présentation. Donc le passage directe de la couche présentation vers la couche accès des données n'est pas permis. Ceci renforce la sécurité des informations.

1.2. Vue composant et connecteur :

Combinaison de 2 styles :

Style architectural: Call return: Client/serveur

Justification : Le système est basé sur des terminaux qui sont connecté à un serveur local et ces serveurs locaux sont connectés à des serveurs centraux. Donc il faut représenter cette structure et documenter la communication entre ces composants. Le style client-serveur peut illustrer l'architecture du système et la communication entre ses composants.

Repository style : shared data style

Justification : Ce style architectural permet de spécifier les répositoires (repositories) de données et les accès à ces données. Les répositoires contiennent les collections de données persistantes. Dans notre cas les données ont des accesseurs multiples, donc il faut documenter et concevoir ces composants et ces connecteurs.

1.3. Vue allocation :

Combinaison de 2 styles:

Style architectural : déploiement.

Justification : Ce style présente le mapping entre les éléments logiciels et les éléments non logiciels ou éléments d'environnement. On aura besoin de pésenter les eléments non logiciels du computing platform (comme les procesors, mémoire, réseau, …). En fait ce style permet d'analyser la performance, la disponibilité et la sécurité de ce système.

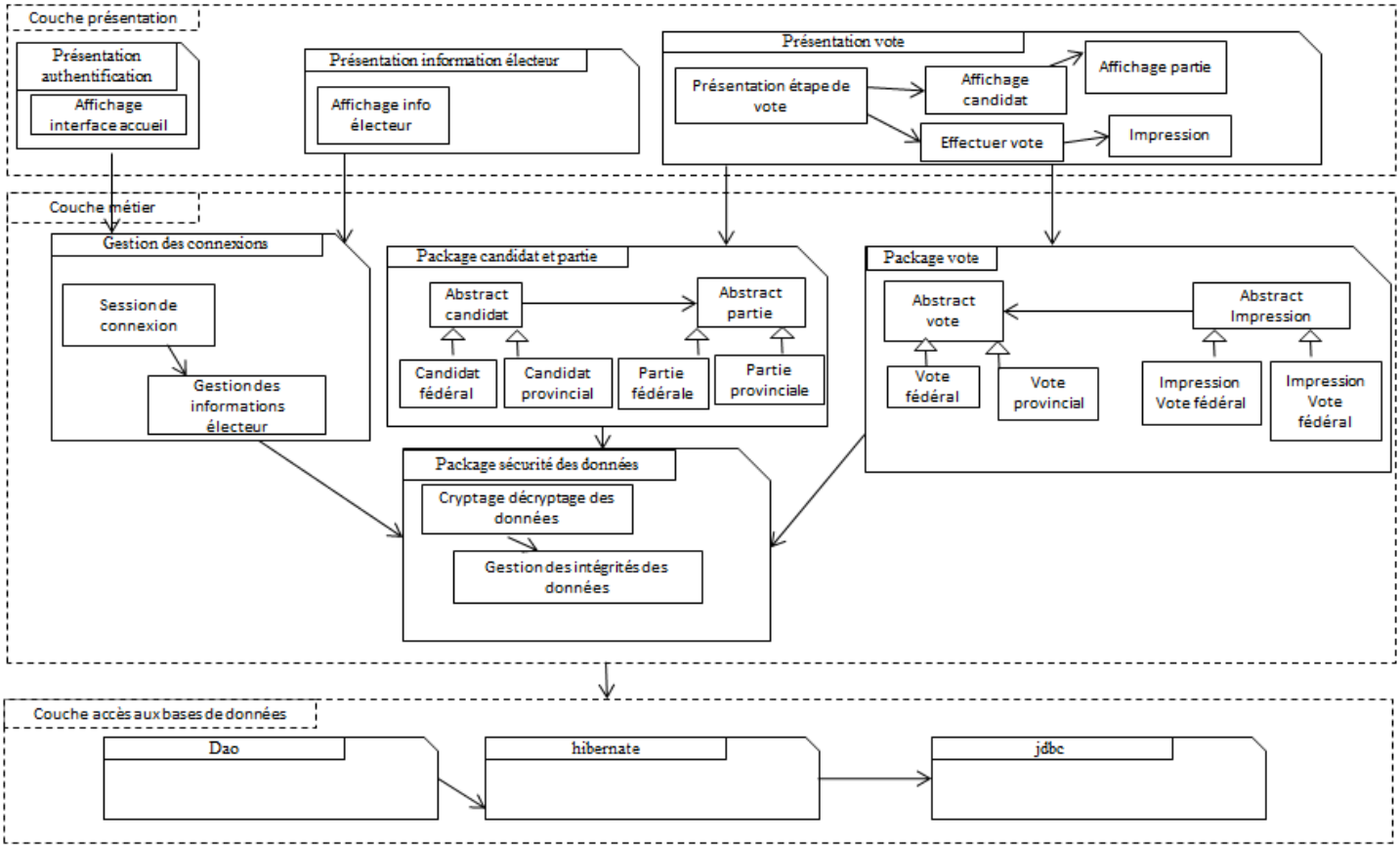
Style architectural : installation.

Justification : Ce style permet de préciser quel composant du style composant et connecteur est alloué à quel file management system dans l'environnement de production. A l'aide de ce style les développeurs, déployeur et agents de maintenance qui vont rejoindre l'équipe auront une idée sur comment le système est organisé (en terme de fichier, répertoire, fichier de config, …)

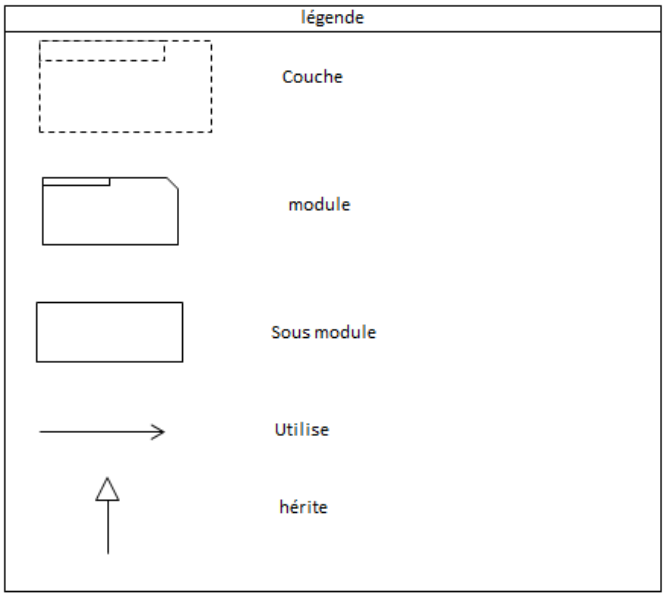
2. Question 2 :

2.1. Vue module

Diagramme :



Légende :



Description :

Cette vue supporte certains attributs de qualité comme la modificabilité et la sécurité.

En effet, la modificabilité est assuré par les différents styles architecturaux utilisés (décomposition, utilise, généralisation, en couche). Pour la décomposition, la séparation des aspects changeables dans un même module facilite la modificabilité. Donc en cas de modification on peut réutiliser les fonctions communes (module connexion, sécurité des données) et juste modifier les modules variables (comme les candidats, les parties). Le style généralisation a supporté en plus la modificabilité par le fait de définir les abstractions des modules comme les modules de vote, impression, candidat et partie. Les aspects communs sont capturés dans le module abstrait et les variations sont spécifiées dans les modules fils. Dans ce cas pour passer d'un système d'élection provincial vers un système d'élection municipale il faut juste ajouter les modules fils qui contiennent les spécificités de ce type d'élection.

Le style utilise permet de faire une claire analyse d'impact en localisant où réside le changement exactement et quel est l'effet de la modification d'un module par rapport aux autres. Le style en couche assure aussi la modificabilité; chaque couche regroupe un ensemble cohérent de service. De ce fait, la modification dans une couche est transparente par rapport aux autres couches du système : elle n'affecte pas les autres couches.

La sécurité est illustrée par le module sécurité des données qui assure le cryptage et décryptage des données et assure la vérification de l'intégrité et la confidentialité des données.

Le style architectural décomposition a permis la séparation des responsabilités et des fonctionnalités (dans chaque couche les modules sont décomposés en des modules

(connexion, candidat, vote) et en plus chacun de ces modules est décomposé en de sous modules ().

Ce qui caractérise le style en couche est que les relations 'allowed to use' sont unidirectionnelles. Donc il n'y a pas de dépendances cycliques entre les couches ce qui facilite la modificabilité et la testabilité. En plus la sécurité est respectée.

Responsabilité des éléments :

Élément	Responsabilités
Couche	Une couche regroupe les modules ayant la même préoccupation et offrant un ensemble de service cohésif
sous module	Un module est décomposé en de sous module ayant plus de services communs
Module	Un module regroupe des fonctionnalités ou des services communs et dans le même contexte
Relation : Utilise	Cette relation présente les relations du style utilise. Un module utilise les services de l'autre module si une relation utilise est définie
Relation : hérite	Cette relation présente la relation de généralisation : elle illustre l'abstraction des services communs. Les classes fils héritent du module mère abstrait
Relation : allowed to use	Cette relation présente la relation entre les couches. Quelle couche peut utiliser quelle couche

Relation attribut de qualité-tactique et vue architecturale :

Modificabilité : Abstract common service : L'abstraction des classes et des services communs entre le système de vote provincial ou le système de vote municipal

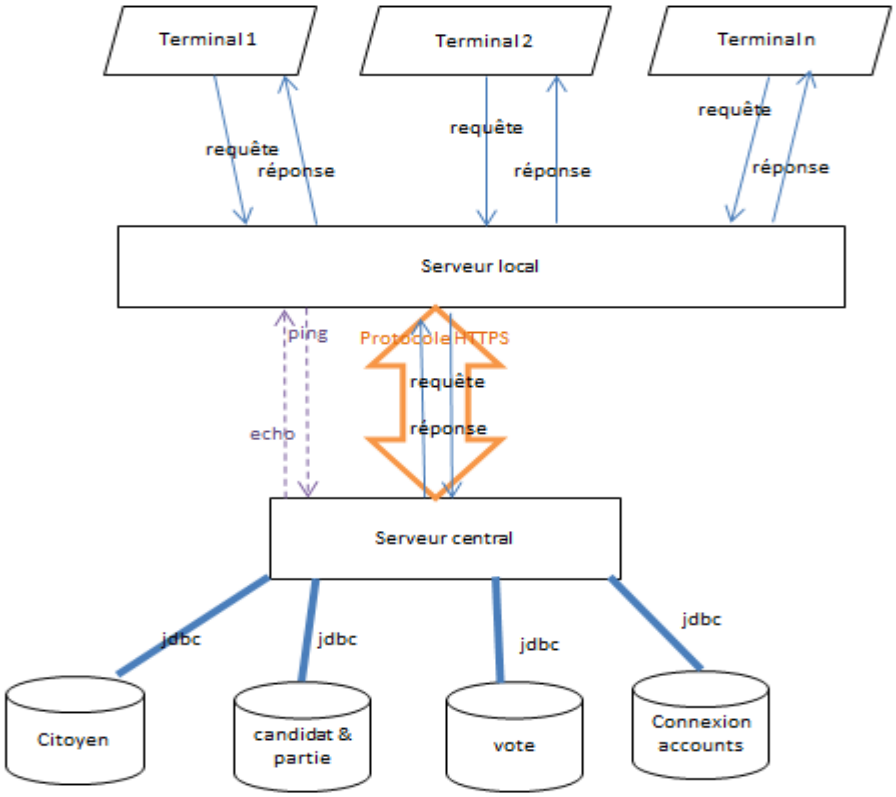
La vue architecturale présente des modules abstraits pour assurer la modificabilité du système comme dans le cas de modification du type d'élection.

Sécurité : Integrité des donneés : La vérification du traffic réseau et la comparaison de toutes les informations qui circulent dans le réseau entre le serveur local et le serveur central.
Pour protéger le réseau et les communications entre les serveurs locaux et le serveur central contre toute tentative d'intrusion au réseau, la vue architecturale propose des modules de sécurités qui gèrent le cryptage et le décryptage de données d'une part et qui vérifient d'autre part si les données transitées ont subies une modification ou non (que ce soit les données concernant l'identité des électeurs et s'ils ont déjà voté ou bien les informations sur le vote enregistrées par l'électeur).

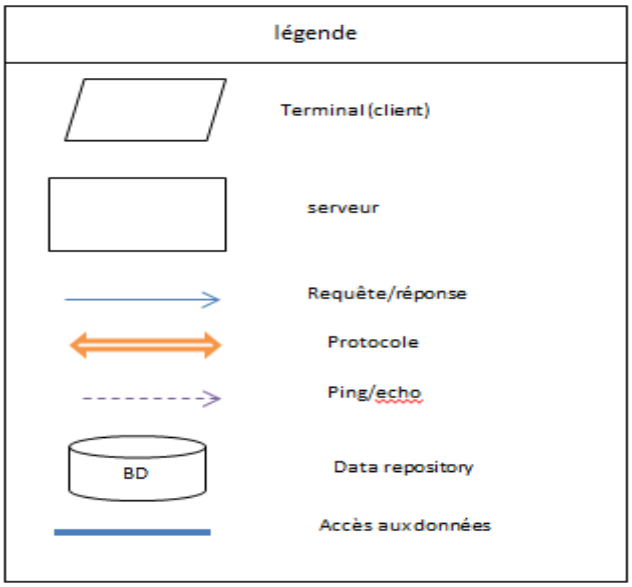
Autres informations : a

2.2. Vue composant et connecteur

Diagramme :



Légende :



Description :

La disponibilité : Pour assurer cet attribut de qualité, nous avons proposé l'usage du ping/echo ; le transfert des messages pour s'assurer du bon fonctionnement des différents composants du système. En effet, le ping/echo va s'occuper de vérifier si le serveur local et le serveur central sont en état de fonctionnement ou pas.

La performance : La solution proposée montre comment le style architectural client/serveur permet d'assurer l'accès concurrent entre les clients (terminaux) pour accéder aux données d'une manière synchrone à travers le service requête/réponse. L'accès concurrent aux serveurs améliore le temps de réponse et donc améliore la performance.
La performance est aussi renforcée par le style shared data style, qui assure la persistance des données qui peuvent avoir des accesseurs multiples et donc l'accès aux données peut être concurrent.
La sécurité : Les protocoles de communication sécurisés (HTTPS). Les informations sont transférées en sécurité en utilisant le protocole http sécurisé (HTTPS) qui transite des données cryptées.

Le style architectural client-serveur montre la structure des terminaux qui sont des clients web légers et les serveurs locaux qui sont des serveurs d'application. Toute communication entre ces deux composants est assurée par les requetes/réponses. Le serveur local joue un double rôle; il devient un client pour le serveur central afin de demander des informations concernant l'état de vote d'un utilisateur ou bien pour avoir les informations sur les candidats et leurs parties et en contrepartie il envoie le résultat de vote d'un électeur et met à jour son état à travers des requêtes envoyées au serveur central.

Le style architectural shared data style, montre la structure des données ainsi que le système de gestion de ces données pour retourner les données percistantes.

Responsabilité des éléments :

Élément	Responsabilités
Terminal (client)	Les terminaux sont considérés comme client léger. Ils communiquent avec le serveur local pour récupérer les informations de l'électeur, de l'interface de vote, des candidats et des parties et pour envoyer les informations de vote.
Serveur local	Ce serveur joue un double rôle : il est considéré comme un serveur pour traiter les requêtes des terminaux et comme un client qui envoie des requêtes au serveur central pour récupérer les informations nécessaires
Serveur central	Le serveur central traite les requêtes des serveurs locaux et renvoie les informations concernant les électeurs et les candidats et reçoit les informations de vote, enregistre le vote et le statut de vote de l'électeur
Requête/réponse	Les requêtes et les réponses sont acheminées entre le client et le serveur : elles contiennent les informations sur l'électeur, le vote, les candidats et les parties
Ping/echo	Message envoyée pour vérifier l'états des connections aux serveurs
Protocole	Le protocole HTTPS est utilisé pour la sécurité des informations transférées entre le serveur local et le serveur central
Data repository	Composant représentant les données distribuées
Accès aux données	Assure la manipulation des données

Relation attribut de qualité-tactique et vue architecturale :

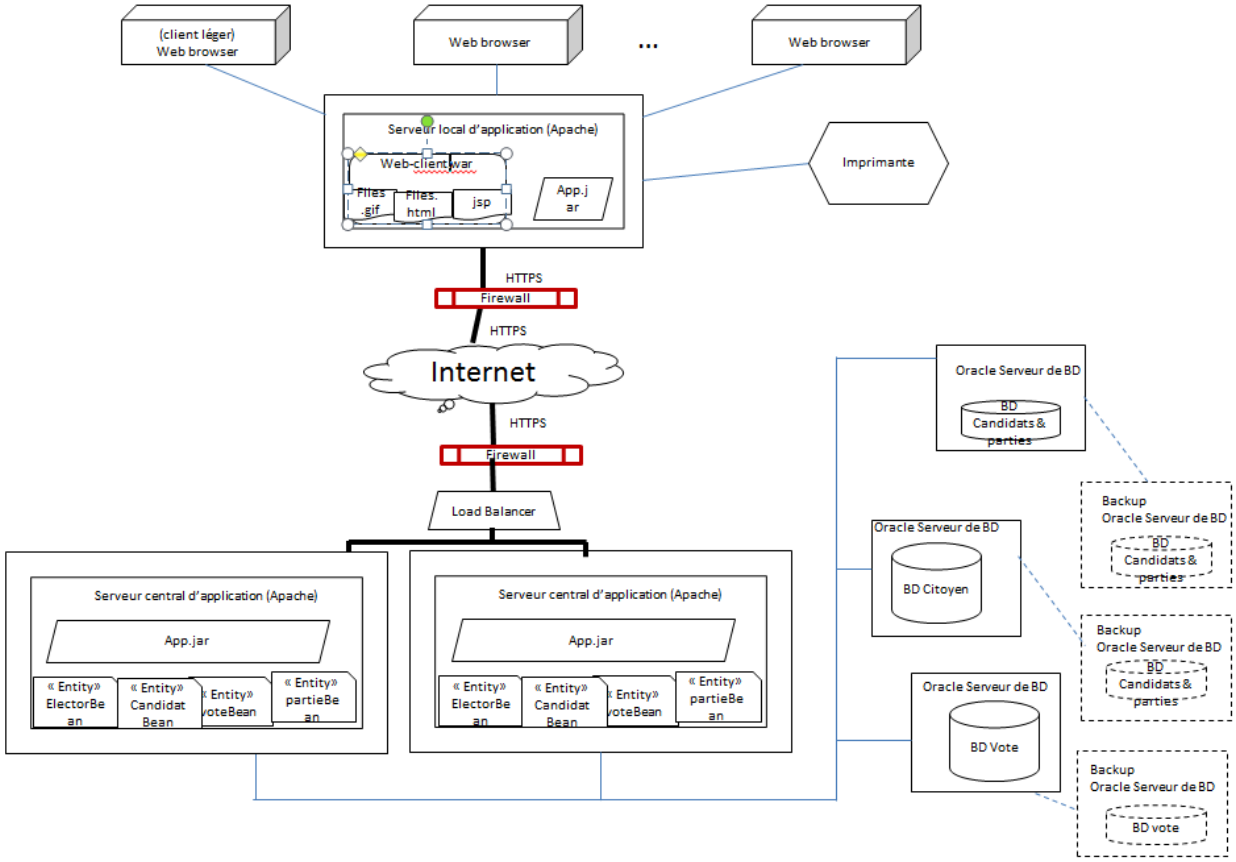
Tactique cryptage des données: les données transférées dans les requêtes et les réponses sont cryptées donc les données sont sécurisées contre toutes attaques ou intrusion.

Disponibilité: Tactique Detect Fault : ping/echo :
L'échange de message ping/echo entre le serveur local et le serveur central vérifie si ces nœuds de réseau sont accessibles et répondent aux ping ou non. Cette tactique est utile pour vérifier l'état de connexion des serveurs locaux et du serveur central et pour s'assurer que la communication entre un serveur local et un serveur central n'est pas interrompue.

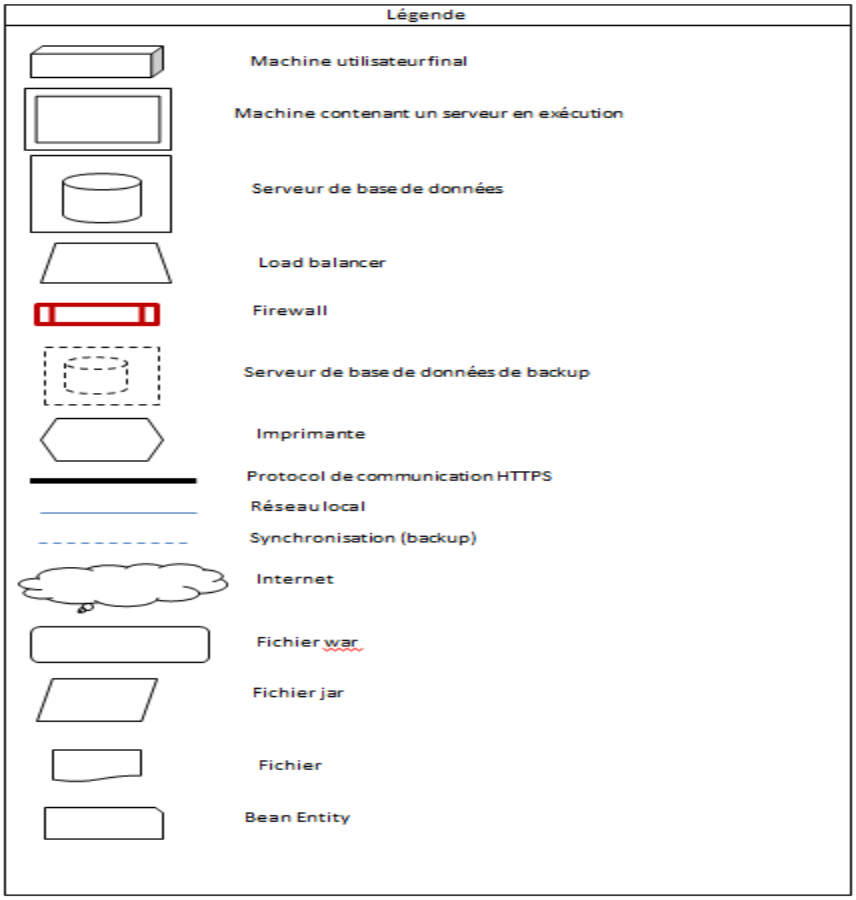
Autres informations : a

2.3. Vue allocation

Diagramme :



Légende :



Description :

Cette vue architecturale respecte certains attributs de qualité tels que la sécurité, la performance et la disponibilité. En effet, la sécurité est assurée par l'usage des firewalls qui séparent les terminaux et le serveur local de toute tentative d'accès non autorisé venant de l'extérieur (internet). Un autre firewall est utile aussi pour protéger le réseau qui contient le serveur central et les serveurs de base de données contre tout accès ou tous flux de données non autorisés. Cette vue illustre aussi l'utilisation du protocole de communication sécurisé HTTPS qui protège les données transitées via le protocole http.

La performance est respectée par l'usage d'un load balancer pour répartir la charge entre les serveurs centraux. En effet en aura plus qu'un serveur central pour accélérer la réponse aux requêtes des serveurs locaux.

Pour assurer la disponibilité, le backup de la base de données va servir pour garder une copie récente des données donc en cas de panne cette copie sera utilisée.

Le style architectural déploiement a permis de schématiser l'organisation et l'allocation des composants (serveur d'application, client web, serveur de base de données, …) aux éléments de l'environnement non logiciels (ou hardware). Il illustre aussi l'organisation du réseau (le réseau local, les firewall, les

protocoles de communication, …). Ce style nous a permis d'analysé la performance, la disponibilité et la sécurité de notre système.

Le style architectural installation décrit comment les éléments du système sont installés. Il précise comment les différents fichier (war, jar, …) sont alloués à quel élément du système pour assurer le bon fonctionnement de l'application dans son environnement de production.

Responsabilité des éléments :

Élément	Responsabilité
Machine utilisateur final	Ce sont les terminaux avec lesquels l'électeur va se connecter pour faire le vote. Elle contient un web browser pour se connecter aux serveurs d'application.
Machine contenant un serveur en exécution	Ce sont les machines qui représentent le serveur local et le serveur central de l'application
Serveur de base de données	Gérer les bases de données
Load balancer	Diviser la charge entre les serveurs centraux pour améliorer la performance
Firewall	Limiter l'accès aux données : prévention contre les flux d'information non autorisés et contre les accès non autorisés qui viennent de l'extérieur
Serveur de base de données de backup	Sert pour le backup de la base de données pour garder une copie récente des données
Imprimante	Pour imprimer les bulletins de vote
Protocole de communication Https	Protocole de communication http sécurisé ou les données transmises sont cryptées
Réseau local	Relie les différentes machines à l'intérieur d'une zone sécurisée
Synchronisation	Pour effectuer le backup (les données sont toujours synchronisées pour garder une version récente des données)
internet	Réseau internet(les machines locaux sont reliées aux machines centraux à travers l'internet)
Fichier war	Archive d'application web pour déployer une application web sur un serveur d'application. Il contient les jsp, les servlets, les pages web statiques (html, javascript, …)
Fichier jar	Archive Java : pour distribuer un ensemble de classe java
Fichier	Les différents fichiers qui peuvent contenir une archive
Bean entity	Composant J2EE coté serveur pour la persistance des données

Relation attribut de qualité-tactique et vue architecturale :

La disponibilité est assurée par Le backup de base de données : avoir une copie de la base de données (toujours synchronisée avec la base de données principale).

La tactique : Maintain multiple copies of computations est manipulée par le fait de dupliquer le serveur central pour réduire la charge sur un seul serveur. Le load Balancer permet de diviser la charge entre les serveurs centraux.

Resist attacks : Encrypt Data : Cette tactique de sécurité est assurée par le protocole de communication sécurisé HTTPS (les données transférées sont cryptées).

Nous avons ainsi augmenté la sécurité par la tactique limit acces en utilisant les Firewalls.