Environnements Logiciels pour l'Informatique Mobile

Synchronisation des applications mobiles

Présentation

- Polytech’Nice-Sophia 2012 (IAM)

- 6 ans chez Sopra-Steria
  - Architecture
  - Projets innovants
  - Formation (Sécurité)

- gregory.marro@soprasteria.com

Sommaire

1. Stocker des données sur le mobile
    1. Sauvegarder des préférences
    2. Écrire un fichier
    3. Gérer une base de données
2. Synchronisation des applications mobiles
    1. Vision globale
    2. Exposer des données pour le mobile
    3. Gérer la désynchronisation
3. TP

Stocker des données sur le mobile

Sauvegarder des préférences

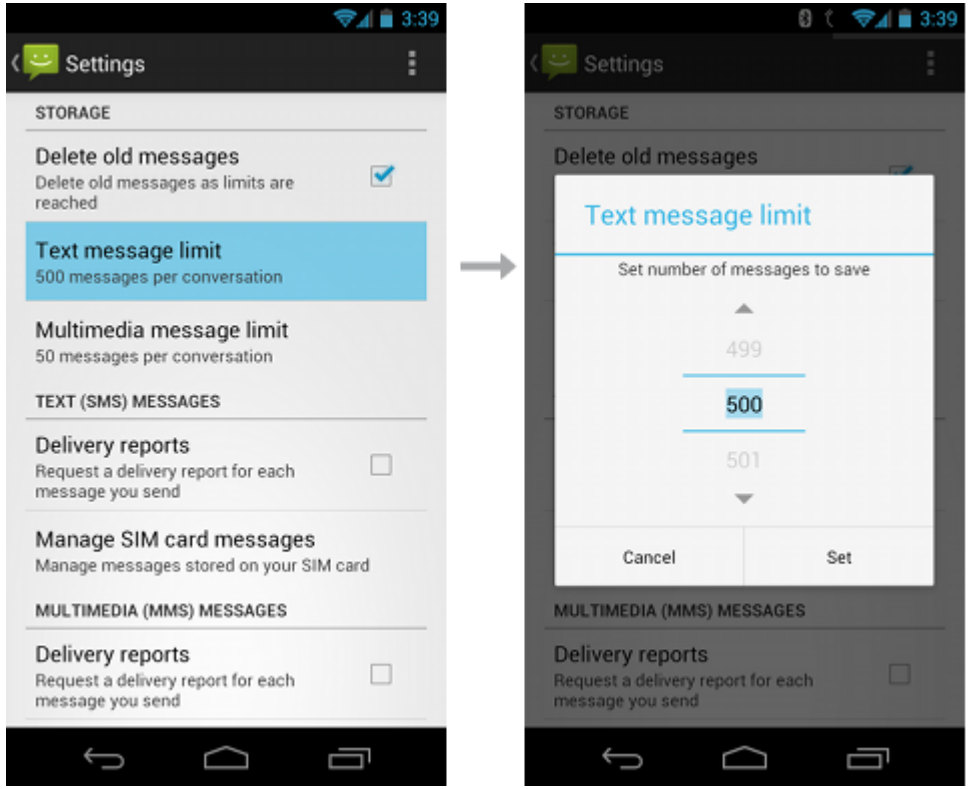
Stocker des données sur le mobile

- De nombreuses applications nécessitent de la persistance de données
  - Paramètres utilisateurs
  - Données de synchronisation
  - Volume de données important à transférer
  - Données persistantes pour l'application

- Plusieurs solutions Android

Sauvegarder des préférences

- Permet de rendre pérenne un petit volume de données
- Généralement utilisé pour les préférences utilisateurs
- Souvent associé aux Settings
- Sous forme clé/valeur



Sauvegarder des préférences

- Utilisation de la classe SharedPreferences

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
```

- Lecture directement de SharedPreferences

```
long highScore = sharedPref.getInt(getString(R.string.saved_high_score), defaultValue);
```

- Modification à l'aide du SharedPreferences.Editor

```
SharedPreferences.Editor editor = sharedPref.edit();
editor.putInt(getString(R.string.saved_high_score), newHighScore);
```

- Ne pas oublier le commit !

```
editor.commit();
```

Stocker des données sur le mobile

Écrire un fichier

Écriture dans les fichiers

- Dans le cas de données volumineuses (par exemple : images, vidéos …)

- 2 solutions :

Interne :
- Toujours disponible
- Uniquement accessible par l'application
- Désinstallation « propre »

Externe :
- Disponible si elle existe et si elle est montée
- Partagé sur tout le téléphone
- Restriction pour la désinstallation
- Permissions requises

Écriture d’un fichier en interne

- 2 types de fichier :
  - Classique : getFilesDir
  - Cache : getCacheDir

```
File file = new File(context.getFilesDir(), filename);
```

- Utilisation de FileOutputStream

- Ne pas oublier le close()

```
outputStream = openFileOutput(filename, Context.MODE_PRIVATE);
outputStream.write(string.getBytes());
outputStream.close();
```

Écriture dans un fichier externe

- Ajout d’une permission au Manifest
- Il faut vérifier que l’on a accès au stockage externe
  - Grace à Environment.getExternalStorageState();
- Puis identique au fichier interne

Stocker des données sur le mobile

Gérer une base de données

Gestion de la base de données

- Utilisation de SQLite (écriture dans un fichier)
- Une base de donnée n’est pas partagée
- Utilisation conseillée de SQLiteOpenHelper
- Dans un Thread séparé

Gestion de la base de données

- Création d'une classe qui hérite de SQLiteOpenHelper

```
public class FeedReaderDbHelper extends SQLiteOpenHelper {
```

- Surcharge des méthodes :

```
public void onCreate(SQLiteDatabase db) {
    db.execSQL(SQL_CREATE_ENTRIES);
}
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // This database is only a cache for online data, so its upgrade policy is
    // to simply to discard the data and start over
    db.execSQL(SQL_DELETE_ENTRIES);
    onCreate(db);
}
public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    onUpgrade(db, oldVersion, newVersion);
}
```

- Utilisation de la base en lecture/écriture

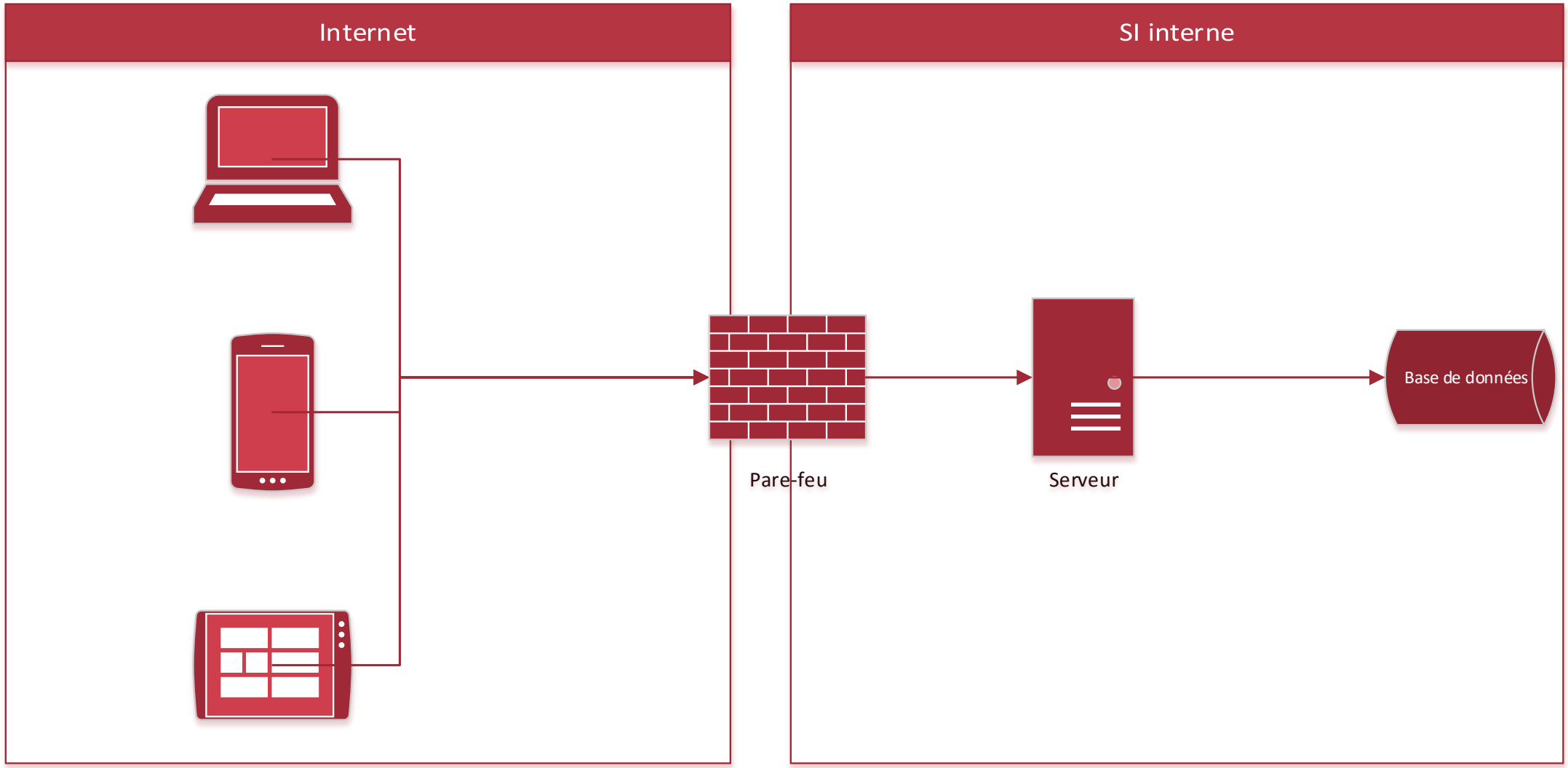
```
SQLiteDatabase db = mDbHelper.getWritableDatabase();

SQLiteDatabase db = mDbHelper.getReadableDatabase();
```

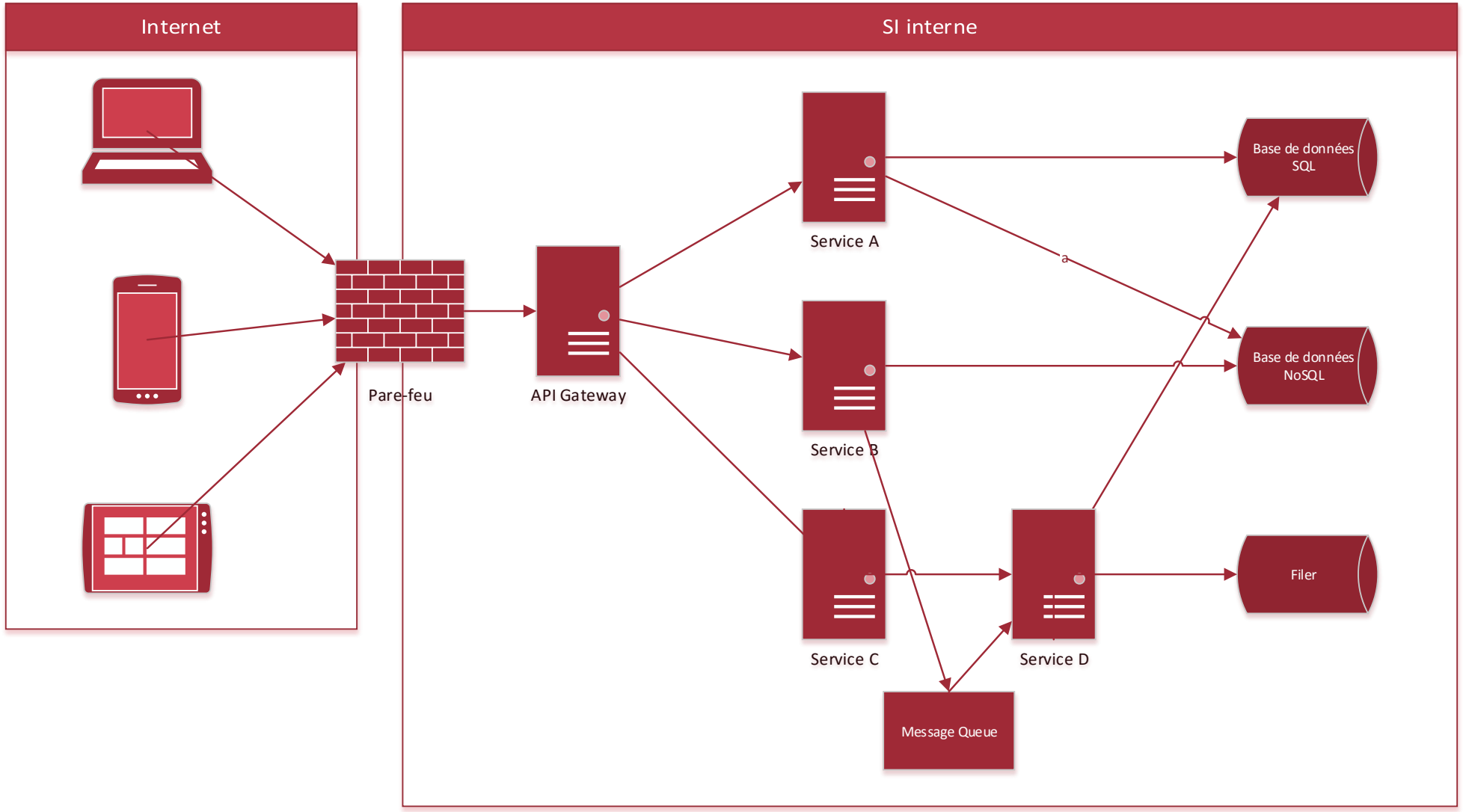
Synchronisation des applications mobiles

Vision globale

Architecture 3-tiers



Architecture microservices



Peu importe l’architecture de l’application,

- Le client a toujours un point d’accès
- Il faut prendre en compte l’existant
- Les clients sont nombreux, attention aux conflits
- Si l’API est ouverte, question de sécurité

Synchronisation des applications mobiles

Exposer des données pour le mobile

RESTful API

- Le REST est à privilégier :
  - Utilisation des méthodes HTTP (GET, POST, PUT, DELETE, PATCH …)
  - URL identifie les ressources
  - Contenu en JSON, XML, CSV … mais JSON de préférence
  - Sans état
  - Jeton d'authentification

Méthodes HTTP

URL	GET	PUT	POST	DELETE
Collection, telle que `http://api.exemple.com/ressources/`	**Liste** les URIs et peut-être d'autres détails des membres de la collection.	**Remplace** la collection entière par une autre collection.	**Crée** une nouvelle entrée dans la collection. L'URI de la nouvelle entrée est assignée automatiquement et est retournée par cette opération[7].	**Supprime** l'entière collection.
Élément, tel que `http://api.exemple.com/ressources/item17`	**Récupérer** une représentation du membre adressé de la collection, exprimé dans un type de média Internet approprié.	**Remplace** le membre adressé de la collection, ou s'il n'existe pas, le **crée**.	Généralement non utilisée. Elle Traite le membre adressé en tant que collection à part entière et **crée** une nouvelle entrée en son sein[7].	**Supprime** le membre adressé de la collection.

Développer un API, quel langage ?

- En fonction du contexte de l'application
- Avec Java & JAX-RS :

```
import javax.ws.rs.core.Response;
import org.json.JSONException;
import org.json.JSONObject;

@Path("/ftocservice")
public class FtoCService {

      @GET
      @Produces("application/json")
      public Response convertFtoC() throws JSONException {
```

- Avec SpringBoot :

```
@RestController
@RequestMapping("/api")
public class RestApiController {

        public static final Logger logger = LoggerFactory.getLogger(RestApiController.class

        @Autowired
        UserService userService; //Service which will do all data retrieval/manipulation wo

        // ------------------Retrieve All Users--------------------------------------------

        @RequestMapping(value = "/user/", method = RequestMethod.GET)
        public ResponseEntity<List<User>> listAllUsers() {
```

Développer un API, quel langage ?

- Avec NodeJS et Express :

```
router.route('/bears')

    // create a bear (accessed at POST http://localhost:8080/api/bears)
    .post(function(req, res) {

        var bear = new Bear();      // create a new instance of the Bear model
        bear.name = req.body.name;  // set the bears name (comes from the request)

        // save the bear and check for errors
        bear.save(function(err) {
```

- Utilisation de LoopBack pour la création d'API RESTful :
  - https://loopback.io

Quelques remarques …

- Prendre en compte le contexte existant
- Limiter le nombre de requêtes au maximum (grappe de données)
- Utilisation de HTTPS systématique
- Utiliser de préférence du JSON pour le traitement sur mobile

Synchronisation des applications mobiles

Gérer la désynchronisation

Pourquoi gérer cette désynchronisation ?

- Dans le cas de gestion d’un mode offline
- Conflit potentiel si plusieurs clients modifient une ressource
- Ne pas négliger cet aspect qui peut représenter une part importante de l’application

Problématique des IDs

- Seul le serveur à le droit de donner un ID
- Le mobile créé des entités avec des ID négatifs
- Une fois la synchronisation faite, on met à jour la base locale avec les nouveaux IDs du serveur

Synchronisation différentielle

- On ne récupère pas toute les données à chaque fois
- Il faut garder un historique des modifications de la base serveur (timestamp, table d'historique…)
- Envoi du timestamp par l'application mobile à la requête de synchronisation

TP : Application météo

- Créer une application qui affiche les prévisions météo dans une liste à partir du point actuel ou d'une ville, si elle a été ajoutée dans les préférences :
  - API OpenWeatherMap : http://openweathermap.org/forecast5#geo5
  - Clé API : c5be641c0caea8bab91e5eaf884bfccc

- Au runtime, récupérer et stocker dans des fichiers des icones pour le temps et les afficher dans la liste en fonction du temps prévu :
  - http://openweathermap.org/weather-conditions#How-to-get-icon-URL

- Stocker en base de données l'historique des choix de villes (geocodés ou entrés par l'utilisateur) pour l'utiliser dans une AutoCompleteTextView