Ce document a été mis en ligne par l'organisme FormaV®.

Toute reproduction, représentation ou diffusion, même partielle, sans autorisation préalable, est strictement interdite.

Pour en savoir plus sur nos formations disponibles, veuillez visiter :
formav.co/explorer

BTS SIO

BTS Services informatiques aux organisations, épreuve E5

Session 2017 • Code sujet : **SI5SLAM** • Durée : **4h00** • Coefficient : **5**

Sommaire

Cliquez sur une partie pour y accéder directement

1. Évolution de l'architecture logicielle (5 questions) **Page 3**

2. Mise en place d'une gestion de stock centralisée (3 questions) **Page 6**

3. Finalisation de l'application de saisie des livraisons de bois (3 questions) **Page 8**

4. Valorisation des déchets (4 questions) **Page 10**

5. Ce qu'il faut retenir & Questions fréquentes (FAQ) (Synthèse & Q&A Examen) **Page 12**

Sujet officiel (PDF) : https://coursbtssio.fr/assets/annales/bts-sio/E5 - Administration des systèmes et des réseaux (spécifique à l'option)/2017/BTS SIO - E5 - Administration des systèmes et des réseaux (spécifique à l'option) - Session 2017 - 1-Sujet.pdf

1. Évolution de l’architecture logicielle

L’association AHM-23 remplace ses fichiers tableur par une application web et une base de données relationnelle hébergées à son siège. La mission porte sur la mise en production, la migration des données, la gestion des accès et la stratégie de sauvegarde.

Question 1.1. Énumérer les composants logiciels types devant être installés pour mettre en production l’application web.

Pour héberger le site et sa base de données au siège, installer les composants nécessaires sur le serveur :

- un système d’exploitation serveur ;
- un serveur web pour recevoir les requêtes HTTP/HTTPS ;
- un serveur d’applications ou un environnement d’exécution adapté à la technologie du site ;
- les bibliothèques et composants requis par l’application ;
- un SGBD relationnel ;
- l’application web et sa configuration.

Les versions doivent être compatibles et les services configurés pour communiquer selon l’architecture retenue.

> **✓ RÉPONSE À RETENIR**
>
> Système d’exploitation serveur, serveur web, environnement d’exécution, application web et SGBD relationnel.

Question 1.2. Présenter les étapes nécessaires à la récupération des données depuis l’application tableur, à l’alimentation de la nouvelle base de données et à la validation de son contenu.

1. Faire une copie de sécurité des classeurs sources et recenser les feuilles et champs à migrer.
2. Nettoyer et harmoniser les données : formats, valeurs manquantes, doublons et références entre feuilles.
3. Exporter les données dans un format d’échange, par exemple CSV, puis les transformer si nécessaire pour respecter le schéma relationnel.
4. Importer les tables dans l’ordre adapté aux clés étrangères.
5. Contrôler les volumes importés, les valeurs, les clés et les relations, puis comparer des échantillons aux sources.
6. Corriger les anomalies et faire valider les résultats par les utilisateurs.

> **✓ RÉPONSE À RETENIR**
>
> Sauvegarder, nettoyer et exporter les classeurs, importer les données, vérifier leur cohérence et faire valider la migration.

Question 1.3. Proposer une solution permettant à chaque chef d'atelier de modifier les informations de son site et au comptable de consulter l'ensemble des informations des deux Esat.

Mettre en place des comptes individuels authentifiés et des rôles associés à un site. Les autorisations doivent être contrôlées côté serveur à chaque consultation ou modification, avec un filtrage des données selon le profil.

Profil	Droits à attribuer
Chef d'atelier	Consulter et modifier les informations de son site uniquement
Comptable	Consulter les informations des deux sites, sans droit de modification si celui-ci n'est pas nécessaire

Cette gestion applique le principe du moindre privilège et évite de se fier uniquement aux contrôles de l'interface.

> ✓ **RÉPONSE À RETENIR**
>
> Comptes authentifiés et droits par rôle : chaque chef modifie uniquement les données de son site ; le comptable consulte les deux sites.

Question 1.4. Présenter au comptable les arguments en faveur du maintien d'une sauvegarde incrémentielle plutôt que différentielle.

Une sauvegarde incrémentielle ne copie que les modifications depuis la sauvegarde précédente. À fréquence égale, elle est généralement plus rapide et consomme moins d'espace qu'une différentielle, qui recopie à chaque fois toutes les modifications depuis la dernière sauvegarde complète.

- Elle réduit le volume transféré et la durée des sauvegardes fréquentes.
- Elle limite l'espace occupé par les sauvegardes au fil du cycle.

Son principal compromis est une restauration plus longue : il faut disposer de la sauvegarde complète et de chaque incrément réalisé ensuite, dans l'ordre.

> ✓ **RÉPONSE À RETENIR**
>
> Les incrémentielles réduisent généralement la durée, le volume transféré et l'espace requis ; leur restauration demande la sauvegarde complète et tous les incréments suivants.

Question 1.5. Indiquer comment utiliser les sauvegardes incrémentielles pour restaurer l'ensemble des données en cas de perte des données de la base.

Pour retrouver l'état des données au moment voulu, restaurer d'abord la sauvegarde complète qui précède cet état. Appliquer ensuite, dans l'ordre chronologique, chacun des fichiers incrémentiels produits depuis cette sauvegarde, sans en omettre.

Après restauration, vérifier l'intégrité de la base et la cohérence des données avant de remettre l'application en service. Il faut donc conserver la sauvegarde complète ainsi que toute la chaîne d'incréments correspondante.

✓ RÉPONSE À RETENIR

Restaurer la sauvegarde complète, puis appliquer dans l’ordre tous les incréments réalisés après celle-ci jusqu’à la date visée.

2. Mise en place d’une gestion de stock centralisée

L’AHM-23 veut connaître le nombre et le coût de revient des palettes fabriquées mais non livrées, puis faire évoluer la base pour intégrer les nouveaux besoins de gestion des modèles et des stocks. Les documents fournissent le schéma relationnel existant et un plan de palette personnalisé.

Question 2.1. Écrire la vue demandée, nommée stockPalettes, avec les colonnes nbPalettes et coutRevientTotal, pour obtenir le nombre total de palettes fabriquées mais non livrées et le coût de revient correspondant. Proposer également comment restreindre l’utilisation de cette vue au seul comptable.

Une ligne est en stock si elle est fabriquée (`estFabrique = 1`) et si sa commande n’a pas de date de livraison réelle. On somme les quantités de ces lignes et leur coût de revient unitaire. Le compte comptable reçoit la lecture de la vue, sans accès direct aux tables ; aucun autre compte ne doit recevoir ce droit.

```
CREATE VIEW stockPalettes AS
SELECT COALESCE(SUM(lc.quantiteCommandee), 0) AS nbPalettes,
       COALESCE(SUM(lc.quantiteCommandee * mp.coutRevient), 0)
           AS coutRevientTotal
FROM LigneCommande lc
JOIN Commande c ON c.id = lc.idCommande
JOIN ModelePalette mp ON mp.id = lc.idModele
WHERE lc.estFabrique = 1
  AND c.dateLivraisonReelle IS NULL;

GRANT SELECT ON nom_base.stockPalettes
TO 'comptable'@'hote_autorise';
```

Remplacer les noms de base, de compte et d’hôte par ceux de l’installation.

> **✓ RÉPONSE À RETENIR**
>
> `stockPalettes` renvoie `nbPalettes` et `coutRevientTotal` ; seul le compte comptable reçoit le droit de lecture sur cette vue.

Question 2.2. Expliquer en quoi la création d’une vue constitue une solution pour répondre au besoin du comptable.

La vue calcule les indicateurs à partir des commandes, de leurs lignes et des modèles, sans créer ni dupliquer de données. Elle rend le résultat disponible sous une forme synthétique et toujours calculée à partir des données présentes.

Elle centralise aussi les critères « fabriquée » et « non livrée », ce qui évite les requêtes divergentes dans plusieurs applications. Le comptable peut consulter les indicateurs sans accéder aux tables détaillées, à condition de limiter ses droits à la lecture de la vue et de protéger les tables sous-jacentes.

> **✓ RÉPONSE À RETENIR**
>
> La vue fournit une synthèse à jour, centralise les critères de calcul et permet de limiter l’accès du comptable aux seules données utiles.

Question 2.3. Proposer une modélisation de l'évolution de la base de données, en intégrant les éléments existants concernés et les nouveaux besoins : modèles standards et personnalisés, clients et contacts, plans et composants, matières premières, pertes et chutes, sites, utilisateurs et stocks par site. Reprendre uniquement les éléments du schéma existant concernés par l'évolution.

Clés primaires indiquées par `PK`, clés étrangères par `FK`. Les champs non cités du schéma existant sont conservés sans changement.

```
Site(idSite PK, nom)
Client(numero PK, raisonSociale, adresseRue, codePostal, ville,
       telephone, courriel, imageLogo, idSite FK)
ContactClient(idContact PK, nom, numeroClient FK)
Commande(id PK, dateCommande, dateLivraisonPrevue,
         dateLivraisonReelle, numeroClient FK)
LigneCommande(idCommande PK/FK, idModele PK/FK,
              quantiteCommandee, prixUnitaireFacture, estFabrique)
EssenceBois(id PK, libelle)
ModelePalette(id PK, designation, longueur, largeur, poidsCharge,
              coutRevient, idEssence FK)
ModeleStandard(idModele PK/FK, referenceCatalogue)
ModelePersonnalise(idModele PK/FK, numeroClient FK,
                   idContactValidateur FK)
PlanModele(idModele PK/FK, fichierPDF)
Composant(idComposant PK, designation, longueur, largeur, epaisseur)
Composition(idModele PK/FK, idComposant PK/FK, quantite)
MatierePremiere(id PK, libelle, longueur, largeur, epaisseur,
                idEssence FK)
Decoupe(idComposant PK/FK, idMatiere PK/FK, quantiteDecoupee,
        longueurPerte, longueurChute)
Utilisateur(id PK, identifiant, motDePasseHash, idSite FK)
StockSite(idSite PK/FK, idMatiere PK/FK, quantiteStockM3,
         stockAlerteM3)
```

Les clés composites de `LigneCommande`, `Composition`, `Decoupe` et `StockSite` identifient leurs associations. L'essence du modèle doit correspondre à celle de ses composants.

> **✓ RÉPONSE À RETENIR**
>
> Modèle proposé : clients et utilisateurs rattachés à un site ; modèles et plans liés à leurs composants ; stocks gérés par site et matière première.

3. Finalisation de l’application de saisie des livraisons de bois

Les chefs d’atelier saisissent les volumes réellement livrés sur des tablettes Android dans des zones sans Wi-Fi, à partir d’une base locale chargée avant leur déplacement. Les documents décrivent les classes, les écrans, le code à compléter et les actions attendues lors de la saisie des volumes.

Question 3.1. Justifier le choix d’une application native pour les tablettes, par rapport aux autres types d’applications.

Les zones de stockage ne sont pas couvertes par le Wi-Fi. L’application native Android est installée sur la tablette et peut donc fonctionner sans connexion au réseau : le chef charge auparavant les données dans la base locale, saisit les volumes sur place, puis les synchronise avec la base centralisée selon le fonctionnement prévu. Une application web classique dépendrait d’une connexion au serveur pour accéder aux données. Le choix natif répond ainsi directement à la contrainte de travail hors ligne et à l’usage d’une tablette Android.

> **✓ RÉPONSE À RETENIR**
>
> Choisir une application native Android : elle permet la saisie hors connexion à partir de la base locale de la tablette.

Question 3.2. En vous inspirant du code du premier écran, compléter à partir de la ligne 5 le code qui alimente le deuxième écran avec les lignes de la commande sélectionnée sur l’écran précédent.

La collection `lesLignes` contient les lignes de la commande sélectionnée, récupérées par la DAO. Pour chaque ligne, on ajoute à `lesDescriptions` la description de sa matière première, fournie par `toString()`. L’adaptateur transmet ensuite ces descriptions à la liste `listeLignes`. L’en-tête reprend le numéro transmis à cet écran, comme sur la maquette.

```
for (LigneCommande uneLigne : lesLignes) {
    lesDescriptions.add(
        uneLigne.getLaMatierePremiere().toString());
}
ArrayAdapter<String> monAdapteur = new ArrayAdapter<String>(
        this, android.R.layout.simple_list_item_1, lesDescriptions);
ListView listeViewLignes =
        (ListView) findViewById(R.id.listeLignes);
TextView textView = new TextView(this);
textView.setText("Contenu commande n° " + numeroCommande);
listeViewLignes.addHeaderView(textView);
listeViewLignes.setAdapter(monAdapteur);
```

> **✓ RÉPONSE À RETENIR**
>
> La liste « listeLignes » affiche les matières premières de la commande sélectionnée, avec son numéro en en-tête.

Question 3.3. Compléter à partir de la ligne 4 le code exécuté au clic du bouton « + » du troisième écran, en suivant les consignes de M. Laplace : ajouter la quantité mesurée à la quantité réelle livrée, afficher la nouvelle quantité totale livrée et mettre à jour la base de données locale.

La méthode `ajoutQuantiteMesuree` cumule le volume saisi avec la quantité réelle déjà livrée. Le champ d'affichage identifié par `totalLivre` reçoit ensuite le nouveau total. Enfin, la DAO enregistre les valeurs mises à jour de l'objet `uneLigneCommande` dans la base locale, conformément aux consignes de M. Laplace.

```
uneLigneCommande.ajoutQuantiteMesuree(quantiteMesureeSaisie);
TextView tvTotalLivre = (TextView) findViewById(R.id.totalLivre);
tvTotalLivre.setText(String.valueOf(
        uneLigneCommande.getQuantiteReelleLivree()));
LigneCommandeDAO.miseAJour(uneLigneCommande);
```

✓ RÉPONSE À RETENIR

Le volume est cumulé, affiché dans « totalLivre », puis enregistré dans la base de données locale.

4. Valorisation des déchets

L'association valorise les déchets de bois et veut suivre le chiffre d'affaires, le poids des commandes et les ventes d'allume-feux par essence sur une période. Les documents décrivent les classes métier et les structures de collection et de dictionnaire nécessaires pour compléter les méthodes demandées.

Question 4.1. Écrire le constructeur de la classe AllumeFeu.

Le constructeur de `AllumeFeu` transmet d'abord à la classe mère les quatre informations communes à tout déchet : identifiant, libellé, poids par sac et prix unitaire. Il initialise ensuite les deux attributs propres à l'allume-feu, le libellé de l'essence et sa valeur énergétique.

```
AllumeFeu(unId, unLibellé, unNbKilosParSac,
          unPrixUnitaire, unLibelEssence,
          uneValeurEnergétique) {
    super(unId, unLibellé, unNbKilosParSac, unPrixUnitaire);
    this.libelEssence = unLibelEssence;
    this.valeurEnergétique = uneValeurEnergétique;
}
```

✓ RÉPONSE À RETENIR

Le constructeur initialise les attributs hérités avec `super`, puis l'essence et sa valeur énergétique.

Question 4.2. Écrire la méthode totalCA de la classe Période.

La collection `lesCommandes` contient les commandes de la période. Le chiffre d'affaires se calcule donc en additionnant le montant total de chacune, puis en retournant la somme.

1. Initialiser le total à zéro.
2. Parcourir les commandes et ajouter leur montant total.
3. Retourner le total calculé.

```
Fonction totalCA() : réel
    total ← 0
    Pour chaque uneCde dans lesCommandes
        total ← total + uneCde.getMontantTotal()
    FinPour
    Retourner total
FinFonction
```

✓ RÉPONSE À RETENIR

`totalCA` retourne la somme des montants totaux des commandes enregistrées pour la période.

Question 4.3. Expliquer ce que retourne la méthode ventesAllumeFeu de la classe Période.

La méthode retourne un dictionnaire dont chaque clé est une essence de bois et la valeur associée est le nombre de sacs d'allume-feux vendus pour cette essence pendant la période.

- Les quantités des commandes sont cumulées par essence.
- Les autres types de déchets ne sont pas comptabilisés.
- Une essence sans vente n'apparaît pas dans le dictionnaire.

> **✓ RÉPONSE À RETENIR**
>
> Un dictionnaire associant chaque essence au nombre total de sacs d'allume-feux vendus sur la période.

Question 4.4. Écrire la méthode getPoidsTotal de la classe Commande.

Pour chaque type de déchet, la ligne indique un nombre de sacs et la clé du dictionnaire donne le poids en kilos d'un sac. Le poids total est la somme des poids de toutes les lignes de la commande.

> Les sacs de sciure, de copeaux et d'allume-feux sont tous pris en compte.

```
Fonction getPoidsTotal() : réel
    poidsTotal ← 0
    lesClés ← lesLignes.donnerLesClés()
    Pour chaque uneClé dans lesClés
        uneLigne ← lesLignes.donnerValeur(uneClé)
        poidsTotal ← poidsTotal +
            uneLigne.getNbSacs() * uneClé.getNbKilosParSac()
    FinPour
    Retourner poidsTotal
FinFonction
```

> **✓ RÉPONSE À RETENIR**
>
> Le poids total, en kilos, est la somme du nombre de sacs de chaque ligne multiplié par le poids d'un sac.

Ce qu'il faut retenir de ce sujet (Session 2017)

Bilan de l'épreuve : Cette épreuve E5 du BTS Services informatiques aux organisations, session 2017, porte le code SI5SLAM et dure 4 heures, coefficient 5. Le cas AHM-23 étudie la migration vers une application web de gestion des stocks, la modélisation de données, le développement Android hors connexion et la valorisation des déchets. Les compétences évaluées couvrent l'architecture, SQL, la conception de bases de données et la programmation.

Notions-clés, formules & règles à maîtriser

- Une vue SQL présente un résultat calculé à partir de tables sans dupliquer les données sources.
- Une sauvegarde incrémentielle conserve les changements depuis la sauvegarde précédente et se restaure après la sauvegarde complète, dans l'ordre des incréments.
- Une clé étrangère relie les données entre tables et contribue à préserver l'intégrité référentielle.
- Le poids d'une commande se calcule en additionnant, pour chaque ligne, le nombre de sacs multiplié par le poids d'un sac.

Points de vigilance & attentes du jury

- Pour la vue de stock, ne retenir que les palettes fabriquées et les commandes sans date de livraison réelle.
- Dans le modèle de données, expliciter les clés, les associations et les attributs nécessaires aux nouveaux besoins.
- Pour le code Android, respecter les objets transmis, les identifiants des champs et les trois actions demandées.
- Pour les méthodes métier, parcourir toutes les lignes et tenir compte des unités et des valeurs définies dans les classes.

Questions fréquentes sur cette épreuve (Q&A Examen)

Q : Comment calculer le poids total d'une commande de déchets ?

Parcourez les lignes de la commande et, pour chacune, multipliez le nombre de sacs par le nombre de kilos par sac du type de déchet. Additionnez les poids obtenus et retournez le total.

Q : Comment restaurer une base avec des sauvegardes incrémentielles ?

Restaurez d'abord la dernière sauvegarde complète, puis appliquez dans l'ordre toutes les sauvegardes incrémentielles réalisées depuis cette sauvegarde. Chaque incrément enregistre les changements depuis la sauvegarde précédente.

Q : Quelle est la durée de l'épreuve et comment les points sont-ils répartis ?

L'épreuve dure 4 heures et compte pour un coefficient 5. Les 100 points sont répartis entre les missions : 20 pour l'architecture, 30 pour la gestion des stocks, 20 pour l'application de saisie et 30 pour la valorisation des déchets.

Q : Quels formats de réponse sont acceptés pour la modélisation et le code ?

Pour la base de données, le sujet accepte un schéma entité-association, un diagramme de classes ou un schéma relationnel. Pour le code de valorisation des déchets, le langage est au choix ou la réponse peut être rédigée en pseudo-code algorithmique.

Sujet officiel associé (PDF) : https://coursbtssio.fr/assets/annales/bts-sio/E5 - Administration des systèmes et des réseaux (spécifique à l'option)/2017/BTS SIO - E5 - Administration des systèmes et des réseaux (spécifique à l'option) - Session 2017 - 1-Sujet.pdf
Ressources BTS Services Informatiques aux Organisations : Toutes les annales & corrigés • Fiches de révision complètes • Simulateur de notes d'examen

Copyright © 2027 FormaV®. Tous droits réservés.

FormaV® est une marque déposée auprès de l'INPI sous le numéro 5227897.

Ce document a été élaboré par FormaV® avec le plus grand soin afin d'accompagner chaque apprenant vers la réussite de ses examens. Son contenu (textes, graphiques, méthodologies, tableaux, exercices, concepts, mises en forme) constitue une œuvre protégée par le droit d'auteur.

Toute copie, partage, reproduction, diffusion ou mise à disposition, même partielle, gratuite ou payante, est strictement interdite sans accord préalable et écrit de FormaV®, conformément aux articles L.111-1 et suivants du Code de la propriété intellectuelle. Dans une logique anti-plagiat, FormaV® se réserve le droit de vérifier toute utilisation illicite, y compris sur les plateformes en ligne ou sites tiers.

En utilisant ce document, vous vous engagez à respecter ces règles et à préserver l'intégrité du travail fourni. La consultation de ce document est strictement personnelle.

Merci de respecter le travail accompli afin de permettre la création continue de ressources pédagogiques fiables et accessibles.