2. **הסבר כיצד תכננתם את הקוד על מנת לספק מודולריות ולתמוך בשימוש חוזר של קוד**

תחילה דאגנו להפרדה הבסיסית ביותר בין המודולים הגדולים ביותר, מסד הנתונים, הממשק הגרפי והלוגיקה. מסד הנתונים הוא שרת mysql 5.6 שאליו אנו מתחברים בעזרת מאגר חיבורים ומבצעים עליו שאילתות שתומכות בכל הלוגיקה של האתר, הממשק הגרפי הם דפי ה- html, jsp (the rendered part) שמספקים את דרך הצגת הנתונים והלוגיקה מאחוריי כל פעולה מתבצעת ב- servlets, jsp (the code snippets). ע"י כך דאגנו כי המודולים הגדולים לא יסתבכו בינהם ויצרו תלויות שאנו לא רוצים. בתוך כל מודול דאגנו להפרדות נוספות, למשל, לכל טבלה יש מחלקה משלה המטפלת בפעולות לה תזדקק הטבלה. בנוסף למחלקות הקיימות לכל טבלה שיש בסכמה שלנו יש לנו מחלקה שמכירה בכל המחלקות הנ"ל על מנת לבצע טרנזקציות, היא מכירה בכל המחלקות והפונקציות שלה המקבלות חיבור ודואגת לקרוא להן לפי הסדר על מנת לבצע את הטרנזקציות.

בנוסף ברמת מסד הנתונים אנו דאגנו לדאוג להפרדה המירבית ביותר בין טבלאות כך שלא תהיינה איזשהן תלויות פונקציונליות בין הטבלאות ודאגנו שלא נשמור מידע מיותר שיכול לגרום לחוסר קונסיסטנטיות במסד הנתונים.

ברמת דפי ה-html אין יותר מדי מה לפרט, דאגנו כי לכל מידע שסביר על הדעת שיהיה באתר יהיה דף html שיציגו.

ברמת הservlets דאגנו תחילה כי לכל פונקציונליות אפשרית בדפי ה- html יהיה לה servlet שיקרא וכך נשמרת המודולריות כך שבעצם כל servlet משרת מטרה אחת בלבד ולא משותף לכמה דפים נוספים. אך ברגע כי שמנו לב כי קיימות חזרה על קוד, למשל כשעוברים לכל עמוד לוודא כי המשתמש אכן מחזיק session חוקי או בעל cookies חוקיים החלטנו לבצע בדיקות אלה בעזרת פילטרים שיבוצעו לכל בקשה וכך בעצם נבטיח חסך בשיכפול קוד ואפשרות להשתמש בפילטר בעוד מקומות.

לסיכום, תמיד היה בראש סדר העדיפויות שלנו לשמור על תכן נקי ומודולרי על מנת להפוך את חיינו לקלים יותר בזמן הניפוי שגיאות וכו'.. עמדנו במשימה היטב ויצרנו קוד בעל תכן מודולרי ומספק.

3. **הסבר כיצד אתם מטפלים בנכונות הקלט ומה עושה את האפליקציה שלכם גמישה ללקוחות שונים**

את הקלט שלנו אנו מעבירים בסופו של דבר למחלקות המתאימות המטפלות בטבלה הרלוונטית לשאילתה. תחילה כל קלט אשר אנו מקבלים מהמשתמש מועבר למסד הנתונים אך ורק דרך preparedStatement על מנת למנוע כל אפשרות של Sql Injection, בכך אנו מבטיחים כי הקלט שהמסד נתונים שלנו יקבל לא יגרום לאיזשהי התנהגות לא צפויה ולא רצויה מצידו וחשיפה של מידע לא נחוץ למשתמש.

הקלט עובר כל הדרך אל המחלקה המטפלת במסד הנתונים הרלוונטי ללא כל בדיקה בדרך, הקלט נבדק אך ורק בפונקציה הרלוונטית במחלקה, תחילה אנו בודקים ערכי null ואם קיימים כאלה אנו זורקים חריגה שתתפס יותר מאוחר ב- servlet המתאים, ולאחר מכן אנו מבצעים את הפעולה המבוקשת ואם במקרה נזרקת חריגה הקשורה לפרמטרים שהמסד קיבל גם אותה אנו זורקים הלאה עד ה- servlet הרלוונטי.

על מנת שהאפליקציה שלנו תתאים ללקוחות השונים המשתמשים בה בכל הדפים אנו לא משתמשים בגדלים הנקבעים בפיקסלים או מדדים כאלה ואחרים, אלא רק במידות המסך, זאת אומרת אחוזים ממנו.

6. **הסבר כיצד האפליקציה שלכם מתמודדת עם שגיאות**

כל השגיאות שיתכנו נזרקות מהמסד הנתונים שלנו לכל פעולה שמתבצעת בכל דף. יצרנו לכל טבלה שיש לנו את כל החריגות היתכנו ממנה ודאגה לכך שה-stackTraceTree של החריגה המקורית יודפס גם בחריגה שלנו על מנת שיהיה נוח לבצע ניפוי שגיאות. בסופו של דבר כל טבלה זורקת חריגות שאנו הגדרנו לפי הצורך שראינו לנכון ומה שקריטי ליצירת חיווי מתאים למשתמש של המערכת. ע"י כך שאנו זורקים מהטבלאות הלאה את החריגות שאנו הגדרנו ויצרנו אנו יודעים כיצד להתמודד עם כל חריגה שיכולה להווצר בכל המודולים המשתמשים במסד הנתונים. ע"י כך אנו בעצם זורקים את החריגות מהמסדים ותופסים אותם ב- servlets הרלוונטיים ובהתאם לכל חריגה אנו יודעים להוציא את הודעת השגיאה המתאימה. אנו גם לא רוצים שחיווי שגיאות תהיה פעולה יקרה שפוגעת בסקלביליות המערכת ומוודאים שכל עמוד מרנדר רק חלק קטן מאוד ממנו ע"י שימוש ב-ajax.