# **ProRecruit – אפיון: תשתית ותחזוקת המערכת**

## **1. ניהול נתונים וגיבויים (Data Management & Backups)**

### **מטרה**

להבטיח שהנתונים במערכת מאובטחים, זמינים וניתנים לשחזור במקרה של תקלה, תוך שמירה על רציפות עסקית ועמידה בתקני פרטיות מחמירים (כמו GDPR).

### **פיצ'רים ופונקציונליות – פירוט מלא**

* **גיבויים אוטומטיים חכמים:**
  + **תדירות:** גיבויים יתבצעו באופן אוטומטי, כאשר גיבוי מלא יתבצע פעם ביום, וגיבויים מצטברים (Incremental Backups) יתבצעו כל מספר שעות.
  + **אחסון:** הגיבויים יאוחסנו בשירות אחסון ענן מאובטח (Google Cloud Storage) תוך שימוש במנגנון הצפנה.
  + **מדיניות שמירה (Retention Policy):** הגיבויים יישמרו לתקופה מוגדרת, כאשר הגיבויים האחרונים יישמרו למשך תקופה ארוכה יותר.
* **הצפנת נתונים מקצה לקצה (End-to-End Encryption):**
  + **בתקשורת (In Transit):** כל התקשורת בין המשתמשים לשרת תהיה מוצפנת באמצעות פרוטוקול **HTTPS/TLS**.
  + **באחסון (At Rest):** נתונים רגישים במסד הנתונים (כמו פרטים אישיים, נתוני שכר) יוצפנו באמצעות אלגוריתמים מתקדמים.
* **נוהל שחזור מהיר (Disaster Recovery):**
  + הגדרה של נוהל פשוט ומהיר לשחזור נתונים במקרה של אובדן, עם בדיקות תקופתיות של הנוהל כדי לוודא שהוא אכן עובד.

## **2. סביבת פיתוח ובדיקות (Development & Testing)**

### **מטרה**

לספק תהליך פיתוח יעיל, מהיר ומאובטח, תוך הקפדה על איכות הקוד ובדיקתו לפני עלייה לסביבת הייצור.

### **פיצ'רים ופונקציונליות – פירוט מלא**

* **סביבות עבודה נפרדות:**
  + **Development:** סביבה מקומית למפתחים, שבה ניתן לבצע שינויים ולבדוק אותם באופן מיידי.
  + **Staging:** סביבת ביניים המשמשת לבדיקות אינטגרציה, בדיקות קצה-לקצה והצגה לצוותי ה-QA והמוצר. סביבה זו תהיה זהה ככל הניתן לסביבת הייצור מבחינת נתונים ותצורה.
  + **Production:** סביבת הייצור החיה, אליה ניגשים המשתמשים.
* **מערך בדיקות אוטומטיות (CI/CD):**
  + **בדיקות יחידה (Unit Tests):** כתיבת בדיקות יחידה לכל רכיב קוד קטן.
  + **בדיקות אינטגרציה (Integration Tests):** בדיקת הקשר והתקשורת בין מודולים שונים.
  + **בדיקות קצה-לקצה (End-to-End Tests):** הדמיית תהליכים מלאים במערכת (למשל, "יצירת משרה והוספת מועמד אליה") באמצעות כלי אוטומציה כמו **Playwright** או **Cypress**.
* **Continuous Integration/Continuous Deployment (CI/CD):**
  + שימוש בשרותי CI/CD (כמו GitHub Actions) כדי להבטיח שבכל פעם שמבוצע שינוי בקוד, בדיקות אוטומטיות מופעלות ורק לאחר שהן עוברות, הקוד משתלב ונפרס אוטומטית לסביבת הבדיקות והייצור.

## **3. מעקב ולוגינג (Monitoring & Logging)**

### **מטרה**

לזהות שגיאות, תקלות ובעיות ביצועים במהירות ופרואקטיביות, על מנת לשמור על יציבות המערכת ולפתור בעיות באופן יזום.

### **פיצ'רים ופונקציונליות – פירוט מלא**

* **מערכת לוגינג מרכזית:**
  + **רישום שגיאות:** כל שגיאה או חריגה במערכת (Backend ו-Frontend) תירשם באופן אוטומטי, יחד עם נתוני הקונטקסט הרלוונטיים (מזהה משתמש, שעת אירוע וכו').
  + **רישום פעולות (Audit Logs):** אירועים קריטיים (כמו כניסת משתמש, שינוי סטטוס מועמד וכו') יירשמו בלוגים, כפי שאפיינו במודול הפיצ'רים החוצים.
* **מערכת ניטור ביצועים (Monitoring):**
  + **ניטור שרתים ומשאבים:** מעקב אחר ביצועי השרתים (CPU, זיכרון, עומס על ה-DB) וביצועי המערכת (זמני תגובה של ה-API) באמצעות כלי ניטור מתקדמים.
  + **ניטור חווית משתמש (User Experience):** מעקב אחר חווית המשתמש בזמן אמת, זיהוי בעיות ביצועים בדפדפן, ושגיאות ב-Frontend.
* **התראות פרואקטיביות:**
  + **הגדרת כללים:** הגדרת כללי התראה שישלחו הודעות למפתחים או לצוות ה-IT במקרה של חריגות.
  + **תצורות:** התראה יכולה להיות "זמן התגובה של API מסוים עלה ב-30% במשך 5 דקות", "יש יותר מ-100 שגיאות ב-Frontend בשעה האחרונה" וכו'.