משרד הדיגיטל הלאומי
רשות התקשוב הממשלתי | gov

הנחיות ראש רשות התקשוב הממשלתי

שם ההנחיה: REST API - מפיתוח לפרסום		
פרק ראשי: ממשל פתוח	מספר הנחיה: 7.2.4	
פרק משני: העברת מידע והנגשת מאגרים	מס' גרסה: 1.0	בתוקף מ-23.2.2021

תוכן עניינים

1. הקדמה 2
2. כללי 2
3. קהל היעד 3
4. מטרת ההנחיה 3
5. הגדרות ומושגים 3
6. שירות API כמוצר 6
7. מתכנון ועד לפרסום שירות API 8
8. סיכום 20
9. קריאה נוספת 21
10. גרסאות ההנחיה 21

1. הקדמה

רשות התקשוב הממשלתי מקדמת מדיניות כוללת לשיפור העברת המידע הממשלתי, מתוך תפיסה הרואה בנתונים כלי משמעותי לייעול עבודת הממשלה ושיפור השירות לאזרחים. על מנת למצות את הערך הגלום במידע ולאפשר שיתוף מהיר, פשוט ויעיל של נתונים, מגבשת הרשות מתודולוגיות וכלים ליישום העברת מידע, הן בפן העסקי והן בפן הטכנולוגי.

במסגרת זו, ההנחיה שלהלן מסדירה את תהליכי העברות המידע באמצעות שירותי API בהיבטי ניהול התהליך ויישומו הטכנולוגי. החל מכניסת הנחיה זו לתוקף, היא מהווה נוהל מחייב לפיתוח ופרסום שירותי REST API במשרדי הממשלה.

2. כללי

רשות התקשוב מקדמת אסטרטגיית העברת מידע בממשלה, במטרה להרחיב את השימוש במידע הממשלתי ככלי לייעול עבודת הממשלה, קבלת החלטות מבוססות נתונים ושיפור השירות לאזרח, בדגש על יישום מדיניות 'שאל פעם אחת' (Ask Once). היבט המרכזי של אסטרטגיה זו הוא קידום שיתוף המידע בין משרדי הממשלה לבין עצמם ובין הממשלה לארגונים מחוץ לממשלה. יישום אסטרטגיה זו מתבסס על מתודולוגיות וכלים ליישום העברות מידע, הן בהיבטים העסקיים והמוצריים (data as a service) והן בהיבטים הטכנולוגיים.

כחלק מאסטרטגיית העברת המידע, הוגדר הצורך במידע איכותי וזמין, הנגיש בזמן אמת (real-time) באמצעות ממשקים מקוונים המאפשרים חיבור ישיר בין מערכות מידע תפעוליות. לעניין זה, הארכיטקטורה המקובלת כיום בעולם התוכנה היא ארכיטקטורת REST, והתקן המקובל לשירותי API בארכיטקטורת REST הוא OAS 3.0.

לצד שיפור היכולת ליישם תהליכי העברת המידע לצורך ספציפי וידוע מראש, פרסום שירותי API מאפשר לארגון להרחיב את יכולתיו העסקיות על ידי מתן מענה לריבוי לקוחות מידע ושימושים והרחבת מעגל צרכני המידע. מימוש פוטנציאל עסקי זה מתאפשר באמצעות פיתוח שירותי API איכותיים ונוחים, העושים שימוש בסטנדרטים אחידים ככל הניתן, ונותנים מענה לתרחישי השימוש הרבים של המידע. אחידות זו תאפשר ללקוחות, בהם משרדי ממשלה אחרים, גופים חיצוניים כגון בנקים, עיריות או מוסדות ציבור לפנות לשימוש בשירות ממשלתי באופן אחיד, תוך פישוט התהליך וקיצור זמני הפיתוח.

לאור התועלות הברורות של סטנדרטיזציה בממשקי העברת מידע, ועל מנת לקדם אחידות ומשילות מידע, הנחיה זו מסדירה את תפיסת ההפעלה, אופן המימוש והסטנדרטים הטכניים לפיתוח ופרסום שירותי API, במטרה לייצר Best Practice שיקל על השיח והשיתוף בין המשרדים וארגונים, ויאפשר שימוש חוזר (re-use) בשירותי API בצורה קלה ופשוטה, תוך חסכון בזמן ומשאבים.

מסמך זה יעסוק בנושאים הבאים:

- השירות כמוצר
- תכנון REST API.
- הנחיות עיצוב ותקן אחיד לשירותים.
- בדיקות השירות.
- פרסום השירות.
- תחזוקה.

3. קהל היעד

- מנהלים ברשות התקשוב הממשלתי.
- מנהלי היחידות מונחות רשות התקשוב במשרדים.
- מנהלי יישומים.
- מנהלי טכנולוגיות.

4. מטרת ההנחיה

להנחות את קהל היעד בדבר מימוש באגפי טד"מ בבואם לתכנן ולפתח שירותי Rest API.

החל מפרסום הנחייה זו, על כל השירותים המפותחים באגפי הטד"מ, או במסגרת מכרזים של חברות חיצוניות, לפעול בהתאם להנחיות המובאות במסמך זה.

הנחיה זו מהווה מסמך מדיניות ותקן לביצוע, ועל כן תוכנה מחייב את קהל היעד.

5. הגדרות ומושגים

5.1 ראה מילון מונחי התקשוב הממשלתי.

5.2 מושגים מתוך ההנחייה

- **Application Programming Interface – API - ממשק תכנות יישומים**

אוסף של פקודות, פונקציות ופרוצדורות מוכנות מראש המאפשרות למתכנתים לעשות שימוש פשוט בהן, מבלי הצורך לכתוב אותן בעצמם. כך יוכלו להשתמש במידע של היישום אותו הם צורכים לטובת היישום שלהם.

- **Representational State Transfer – REST**

ארכיטקטורה למימוש שירותי רשת המבוססים על "שרת-לקוח". לקוחות יוזמים בקשות לשרתים, השרתים מעבדים את הבקשות ומחזירים תגובות מתאימות. בשיטת עבודה זו, הלקוח יכול להיות בשני מצבים: "מצב מנותק" – מצב בו הוא מתקשר עם המשתמש ולכן לא תופס משאבים מהשרת, ומצב "שינוי מצב" - בו השרת מעבד את הבקשה של הלקוח עד החזרת תגובה.

- **שירות API**

שירות ממשק תכנות יישומיים הינו המוצר שמשמש את עולם מערכות המידע להרחבת היכולות העסקיות שלו מחוץ לגבולות הארגון, לצריכה ועדכון של מידע וחשיפת מידע קיים.

- **צורך השירות**

מפתח שמעוניין לפנות לשירות API על מנת לשלוף מידע או לעדכן מידע, כחלק מתהליך עסקי הנמצא בפיתוח. צרכן השירות יכול לאתר שירותים קיימים בקטלוג שירותים.

- **מפרסם השירות**

מפתח תוכנה שפיתח שירות API הנותן מענה לצורך עסקי לשיתוף מידע או עדכון מידע. בסיום תהליך הפיתוח השירות מתפרסם בקטלוג שירותים.

- **קטלוג שירותים**

קטלוג בו מפורסמים השירותים שניתנים לשימוש על ידי צרכנים. בקטלוג ניתן לאתר שירות בהתאם למילות מפתח, להירשם לשירות ולקבל מידע על השירות.

- **שדרת המידע**

תשתית ממשלתית מאובטחת לשיתוף והעברת מידע בין משרדי ממשלה ובין הממשלה לארגונים חיצוניים.

- **HTTP Status Code**

כל תשובת HTTP מלווה בשדה קוד המציין את התוצאה של השרת לנסות למלא אחר הבקשה שנשלחה. על פי שדה זה ניתן לדעת האם הבקשה הצליחה או נכשלה וכיצד להמשיך בהתאם.

- **RESTful Web Service**

Web Service המיושם באמצעות שימוש ב HTTP ובעקרונות REST.

- **Inner API**

שירותים פנים ארגוניים המשמשים את המערכות הפנים משרדיות השונות. לרוב, המינוח המקובל של פניה בין שירותים בתוך ארגון הינו תעבורה east – west.

- **Outer API**

שירותים המתפרסמים בשדרת המידע ומיועדים לשימוש לקוחות חיצוניים של הארגון. API זה יתבסס על inner API ובפועל "יחצין" inner API שנכתב לשימוש פנימי. לרוב, מינוח מקובל של פניה לשירות חיצוני אשר פונה לשירות פנימי להשלמת תהליך הפניה למידע הינו north – south.

- **API Gateway**

שכבת תיווך בין השירותים החיצוניים לשירותים הפנימיים. משמשת ל"תרגום" בין הממשקים, שמירת לוגים, ניהול האבטחה וכו'.

- **Technical Contract**

"חוזה" בין הצרכן למפרסם השירות בו מוגדר מה השירות עושה, מה הן הפונקציות שלו, כיצד לפנות אליהן בצורה נכונה, איך חוזר המידע, אילו שגיאות צפויות וכו'.

- **Design First**

יצירת ה API ובפרט ה Technical Contract, בה מייצרים קודם את "חתימת" הפונקציות השונות, ורק לאחר מכן מייצרים את הקוד ושיטות העבודה שמאחוריו.

- **Code First**

יצירת ה API ובפרט ה Technical Contract, על בסיס קוד קיים והנתונים אותם הוא מחזיר.

- **eXtensible Markup Language – XML**

פורמט טקסטואלי להעברת נתונים בין מערכות שונות, הפועלות על גבי תשתיות שונות. תקן זה נקבע ע"י ה W3C ומתאפיין בשמירת תיאור הנתונים עם הנתונים עצמם, סידור נתונים בצורה היררכית, וכן שימוש בתגיות המגדירות את רכיבי המידע השונים.

- **JavaScript Object Notaion – JSON**

פורמט טקסטואלי המיועד להעברת מבני מידע המורכבים מזוגות של מפתחות וערכים. משמש בעיקר להעברת מידע בין שרת ללקוח כתחליף לפורמט XML. שיטה זו מאופיינת בקלות הקריאה שלה ע"י בן אדם, וכן בהקטנת כמות המידע הפחות רלוונטי שעוברת בו, כגון תגיות.

- **OpenAPI Specification – OAS 3.0**
- תקן המגדיר כיצד ליצור ולצרוך RESTful APIs בלי הצורך בקריאת קוד המקור, מסמכים מיוחדים או ניתוח תעבורת הרשת.

- **Token**

מזהה ייחודי המשמש להזדהות ואימות המשתמש איתו פונים לשירות.

- **Distributed Denial Of Service – DDOS**

שיטת התקפה על שירותי רשת בהם פונים לשירותים ממספר גדול של מקורות ומציפים את רוחב הפס של השרת המארח את השירות. ייתכן והתצפה נגרמת בעקבות ריבוי פניות וייתכן והיא נגרמת בעקבות פניות עם הרבה מידע שאמור לחזור ללקוח.

- **Swagger**

מסמך תיעוד המתאר את השירות הכתוב JSON או YAML. כל שירות המפרסם מחייב פרסום קובץ תיעוד במקביל. swaggerקובץ זה ישמש את הפורטל להצגת מידע על השירות וישמש מוצרי אבטחת מידע לצורך הקשחת השירות (מה מותר בתעבורה בזמן פניה ומה אסור).

- **YAML Ain't Markup Language - YAML**

הוא פורמט סריאליזציה טקסטואלי שנועד להיות קריא בקלות לבן אנוש ,(human readable) המשמש בדרך כלל לקובצי תצורה (קונפיגורציה) וביישומים בהם מאוחסן או מועבר מידע .

6. שירות API כמוצר

השימוש בשירותים מיועד לאפשר שיתוף ועדכון מידע בין ארגונים.

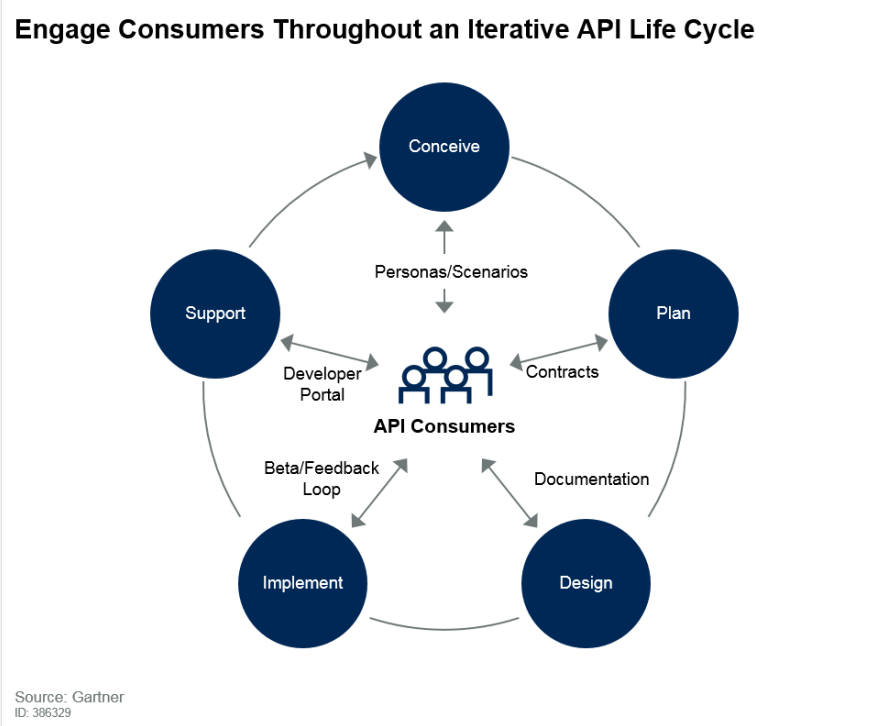
במציאות של שירותים המפותחים לשיתוף מידע, יש להתייחס ללקוחות השירות כאל משתמשים (הם צרכני השירות בפועל), כפי מתייחסים ללקוחות של מערכות מידע. בהתאם לתפיסה זו, מובנת האמירה שיש להתייחס לשירות כאל מוצר בעל מחזור חיים של ייזום, תכנון, אפיון, פיתוח, בדיקות, ייצוב, תמיכה ותחזוקה.

- יש להגדיר באופן ברור את מטרת השירות שיש לחשוף אל מול הצורך העסקי.
- יש לאפיין שירות נוח וברור לצרכן השירות (בדומה למערכות מידע בעלות UI/UX נוחים למשתמש), על מנת להבטיח קלות השימוש בשירות.
- ניהול מחזור חיים של שירות החל משלב הייזום והגדרת הצורך העסקי, אפיון ותכנון, פיתוח ובדיקת השירות, ייצוב ועד לתחזוקה שוטפת וניהול שינויים, כולל זיהוי הלקוחות של השירות, ומה הצורך העסקי שהשירות נותן לו מענה (לקוח פנים ארגוני, ממשרד ממשלתי, ממוסד עסקי או לקוח מהמגזר הפרטי).

6.1 מטרות ויישום

מטרה	יישום
השירות מפותח בתפיסה שמתחשבת במשתמש שעתיד לעשות שימוש בו	השירות מפורסם עם תיעוד, דוגמאות קוד והסברים מקיפים
השירות חושף את כל המידע שמוגדר בתיחום שלו מבלי לחשוף את המבנה הפנימי הארגוני	השירות מרכז בו את כל הפניות לשירותים פנימיים או בסיסי נתונים פנימיים באופן שקוף למשתמש
משתמש שמעוניין להשתמש בשירות ימצא אותו בקלות	הקפדה על תיעוד ומילות מפתח לאיתור השירות בשלב פרסום השירות בפורטל
מאפשר שימוש נוח ומהיר בכל יכולות	השירותהפונקציונליות החשופה בשירות, אחידה לאורך כל השימוש בשירות

בשרטוט הבא ניתן לראות את שלבי מחזור החיים של שירות API בהתייחס למרכיבים השונים.



במעגל החיצוני העיגולים מציינים את מחזור החיים.

בתוך המעגל יש התייחסות למרכיבים שצרכני השירות נזקקים להם בעת צריכת השירות:

- הצורך העסקי
- חוזה השירות
- תיעוד השירות
- התנסות בשירות
- פורטל מפתחים

7. מתכנון ועד לפרסום שירות API

7.1 כללי

השימוש בשירות מאפשר שליחת בקשה (HTTP Request) וקבלת מענה (HTTP Response). שירות API יכול לשמש כל אתר או אפליקציה ללא תלות בשפת הפיתוח, מכיוון שהבקשות מתבססות על הפרוטוקול האוניברסלי של HTTP. המידע מוחזר בפורמט של JSON או XML המוכרות כפורמט לתקשורת, בעדיפות ל-JSON.

חשיפת שירות API מאפשרת גישה למערכות חדשות וישנות (Legacy Applications), למידע ולפונקציונליות. שימוש חוזר בלוגיקה עסקית (עקרון ה- reuse), מאפשר הזדמנויות עסקיות חדשות ושיתוף מידע בין ארגונים.

ככל שה- API קל יותר להבנה על ידי המפתח שמעוניין להשתמש בו, כך זמן הפיתוח שלו יצטמצם ויקצר את זמן הפיתוח סביבו.

לפיתוח ממשקים יישומיים פנים-ארגוניים (Inner API), זוהי נקודה עקרונית וחשובה. משמעות הדבר הינו שככל שיש API ברור לשימוש ואשר נותן מענה לצרכים עסקיים רבים, תהיה פחות גישה ישירה לבסיס הנתונים או לממשקים בצד השרת. ניקח לדוגמא משרד שיש בו מאגר מידע שמחזיק מידע בעל ערך עבור מערכות רבות פנים ארגוניות. ככל שהצוות המנהל את מאגר המידע יחשוף שירותי API נוחים לשימוש שנותנים מענה לצרכים עסקיים, כך יימנע הצורך של מפתח מצוות אחר לתשאל את בסיס הנתונים באופן ישיר.

נרחיב את הדוגמא, במידה ובמאגר המידע המדובר קיים מידע שמעניין גורמים ממשרדים נוספים, אין למפתח ממשרד חיצוני יכולת לפנות ישירות לבסיס הנתונים לביצוע השאילתא. במקרה הזה, המפתח ממשרד חיצוני יצור קשר עם המשרד בעל המידע ויבקש לצרוך שירות שיתן לו מענה לצורך העסקי (outer API). במידה והשירות מפורסם בקטלוג שירותים בצורה טובה, המפתח יוכל לאתר שפורסם שירות שנותן לו מענה לצורך העסקי, לבחון את השימוש, לקרוא את המידע על השירות ולאחר בחינת השירות להירשם לשירות.

7.1.1 סוגי API

סוג ה- APIs	פירוט
פנימיים, Inner APIs	שירותים אלו חושפים ממשקים לשימוש פנימי של מערכת. מערכות אלו יכולות להיות שירותים מסוגים שונים: mini-service, micro-service, macro-service, legacy system service
חיצוניים, Outer APIs	שירותים אלו מיועדים לשימוש של לקוח חיצוני, יושבים על גבי ה– Inner Api's הפנימיים, ובנויים מראש לשימוש של לקוח חיצוני שפונה לקבלת מידע.
מתווכים, API mediation (API Gateway)	שכבה זו מייצרת את הקשר בין שכבת הממשקים הפנימיים (Inner API's) לבין הממשקים החיצוניים (Outer APIs). שכבה זו מנהלת את הלוגים, האבטחה, ההזדהות והתקשורת של כל הפניות. שכבה זו יכולה גם לתרגם את הפניות בין השכבה החיצונית לשכבה הפנימית.

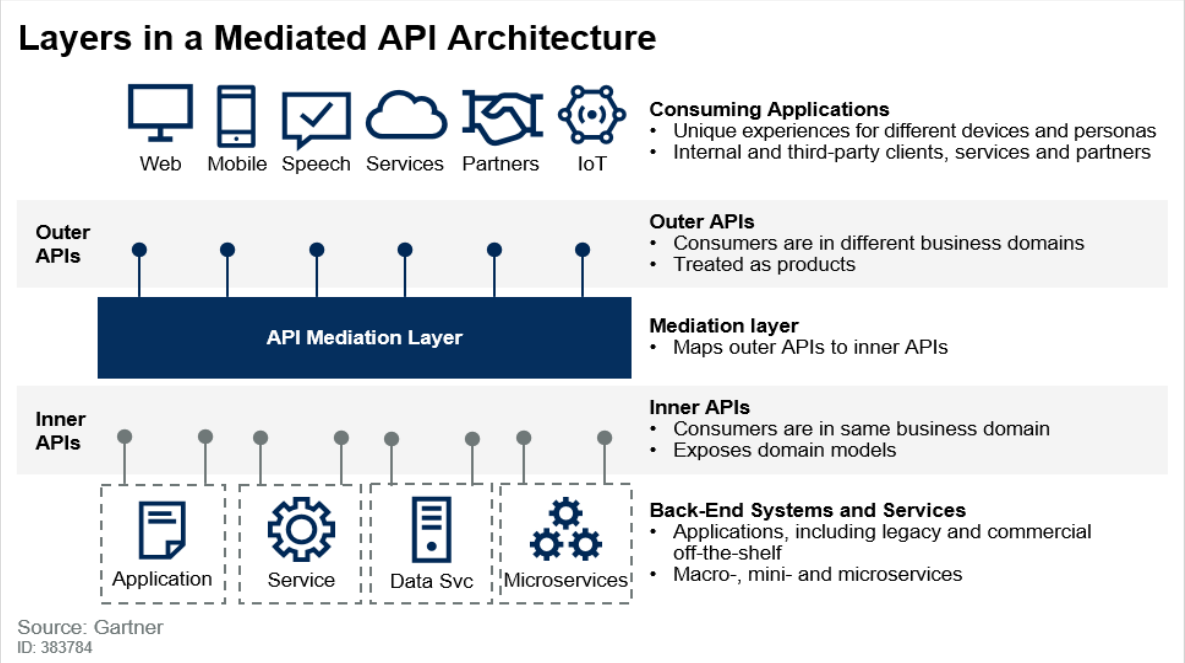
בתרשים הבא ניתן לראות המחשה של השכבות השונות.

- שכבת Consuming Applications

מערכות מסוגים שונים למשתמשים שונים עבור צרכים מגוונים. ללא הבדל בסוג המשתמש וסוג המערכת הפונה, הפניה לשירות API תהיה זהה ותתבצע פניה ל- API החיצוני שמפורסם על ידי בעל המידע.

- שכבת Outer API
- API Mediation Layer
- Inner API
- Back-end systems and services

כל המערכות, שירותים ובסיסי נתונים הנמצאים ברשת הפנימית שנותנים מענה לשירותים המבקשים לשלוף מידע, לבצע תהליך עסקי, לעדכן מידע וכד'.



7.1.2 Data Formats

סוג הפורמט	פירוט
JSON	שפה זו נגזרת משפת Javascript. היא תומכת בשילוב של name/value ורשימות. שפה זו נמצאת בשימוש נרחב בשנים האחרונות, היות שהיא קלת משקל ומשתלבת היטב עם דפדפנים ומכשירים ניידים. **ככלל, תהיה העדפה לשימוש ב- JSON.**
XML	שפה זו היתה נפוצה בשימוש של שירותים מסוג SOAP. בשל התגיות הקיימות בשפה זו, היא כבדה יותר לשימוש, מחייבת PARSING מסובך יותר.

7.2 תכנון שירות API

תכנון שירות דומה לתכנון מערכת מידע. יש להשקיע זמן וחשיבה על הצורך העסקי ואופן המימוש הטוב ביותר. פיתוח שירות ללא תכנון או ללא צורך עסקי ברור, עלול להביא למצב שאין נרשמים לשימוש בשירות, או שהשירות בעל שימושיות נמוכה.

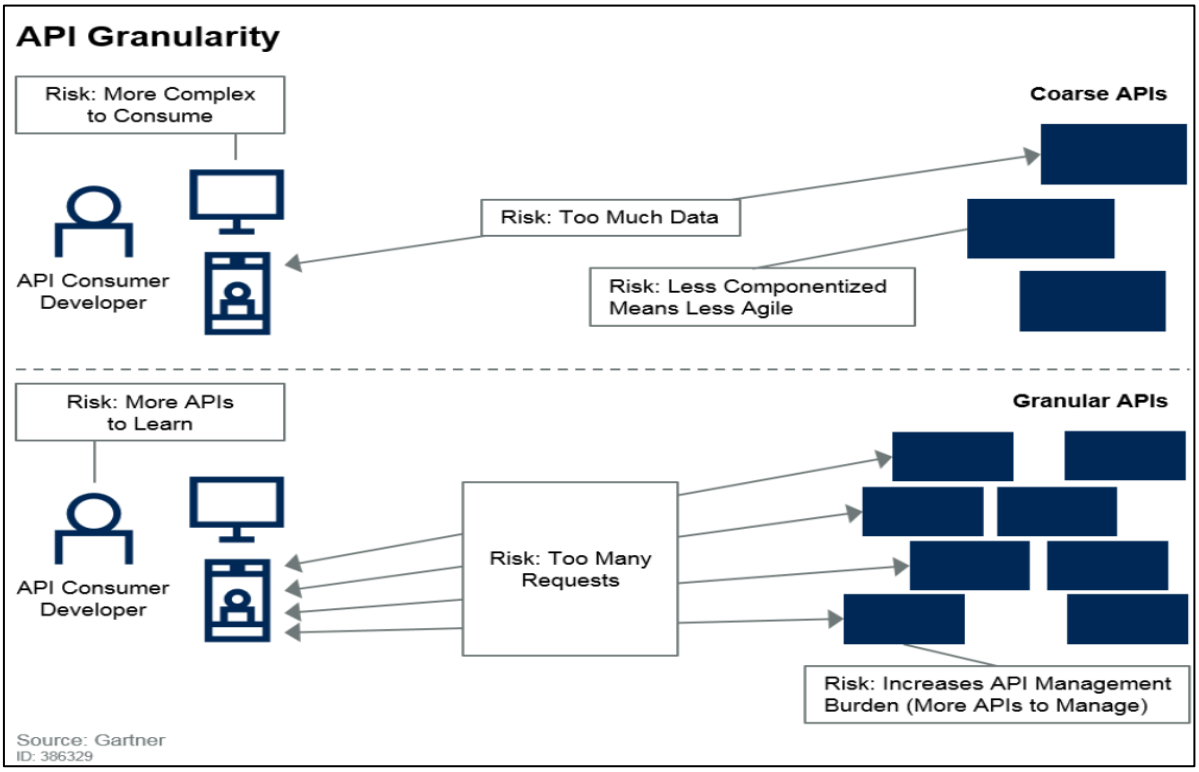
השימושים הנרחבים ביותר בשירותים, הינם למערכות web, מערכות mobile ו- service to service. עבור מערכות web ו- mobile ניתן יהיה לזהות צרכים שונים. לדוגמא, במערכת web ייתכן שהיקף המידע שיידרש משירות יהיה רחב יותר מאשר מערכות mobile שבהן נרצה להציג פחות מידע.

במערכת legacy ייתכן שתהיה דרישה לפיתוח שירות שיאפשר גישה למערכת עצמה. מערכות ה - leagacy הן לרוב מערכות מונוליטיות סגורות שלא ניתן לפנות אליהן ממערכת חיצונית. פיתוח שירות API שיכול לפנות למערכת מונוליטית על מנת לשלוף או לעדכן מידע, פותח את המערכת להזדמנויות ושימושים חדשים.

השקעה בתכנון שירותים בהתאם לצרכים הקיימים, תאפשר בעתיד גישה קלה ותחזוקה נוחה לצרכים חדשים שיעלו. בשלב תכנון השירות יש לבחון את רמת המורכבות שלו והאם יש צורך לפצל אותו לשירותים נוספים או להישאר בשירות עם היקף רחב. לכל תצורה יש שיקולים שונים ולכן יש לבחון את השימוש מול הצורך.

נחלק את עולם השירותים ל -2 חלקים עיקריים:

- שירותים בעלי היקף רחב יותר (Coarse-grained) מכילים יותר פונקציונליות ולכן כל שינוי הינו בעל השפעה רחבה.
- שירותים בעלי היקף צר (Fine-grained) מכילים פונקציונליות ממוקדת, אך בשל ההיקף הצר יהיו יותר שירותים דבר שיגרום ליותר תחזוקה. ככל שהשירותים הינם צרים בהיקף המידע שהם מחזירים, כך יידרשו פניות רבות יותר למספר שירותים רב על מנת לקבל את כל המידע שהלקוח צריך. שירותים מסוג זה יקשו יותר על הצד הצורך, היות ויידרשו יותר בדיקות (עבור כל שירות) ויהיה צורך בהיכרות ואינטגרציה לשירותים רבים יותר.



7.3 עיצוב שירות API (API Design)

7.3.1 חוזה טכני Technical Contract

שירות מהווה נקודת תקשורת בין מפתח השירות לבין צרכן השירות. על מנת לייצר שיח פתוח וברור בין הצדדים, חשוב מאד לבנות חוזה טכנישמגדיר בצורה ברורה את הציפיות ואת אופן השימוש בשירות. ניתן להתייחס לחוזה הטכני כאל "מדריך למשתמש" שמפרט את כל תכולות השירות.

אופן יצירת מסמך התיעוד (swagger) יכול להיות בהתאם לסוגי תפיסה שונים:

- Design first

  כתיבה ב- json או yaml במוצר קוד פתוח swagger editor (https://swagger.io/tools/swagger-editor/)

- Code first

  יצירת ה- swagger מתוך הקוד (generated swagger). ניתן לבחון את הכלים הקיימים שבאמצעותם ניתן לבצע את היצירה מתוך הקוד במאמר https://swagger.io/tools/open-source/open-source-integrations/.

בחוזה יש להחליט על מספר עקרונות שלפיהם השירות יפורסם:

עקרון	הסבר	השאלה שעליה עונה העיקרון
Interface models	הפעולות והישויות אותן השירות מתחייב לספק לצרכן. אילו פונקציות קיימות, מה מבנה הישויות וכו'.	איזה מידע מועבר מהשירות?
Data formats	הדרך בה יאורגן ויקודד המידע בהעברתו כהודעה לשירות (לדוגמא XML או JSON)	מהו אופן העברת המידע - JSON or XML?
Message exchange patterns (MEPs)	כיצד תתבצע פניה לשירות, האם בצורה סינכרונית (בה ממתינים להודעת תשובה לאחר בקשה) או א-סינכרונית (בה ממשיכים בתהליכים עד אשר מתקבלת הודעת תשובה)	מהי שיטת העברת המידע ? סינכרוני – request/response או א-סינכרוני - Publish-subscribe
Messaging protocols	פרוטוקול העברת המידע בין הצרכן לשירות, למשל HTTP או WebSockets	באיזה פרוטוקול תקשורת ניתן להתקשר עם השירות?
Granularity	היקף פירוט הנתונים שהשירות מעניק. האם מדובר בהרבה שדות שחוזרים בעבור השאילתה, או רק בשדות אחדים העונים על הדרישה המינימאלית	מהו היקף המידע והפונקציונליות שהשירות מספק?
Authentication	כיצד מתבצע תהליך אימות הזהות של המשתמש מול השירות.	מהי שיטת האימות?
Exception handling	אילו מנגנונים מופעלים במקרה ומתרחשת שגיאה בשירות? אילו שגיאות חוזרות? האם נשמור את המידע שמכילה השגיאה?	ניהול שגיאות - באיזה אופן השירות מגיב לשגיאות?

7.3.2 תיעוד (Documentation)

יש להיצמד להגדרות כפי שהן מוגדרות ב - OpenAPI 3 Specification.

על קובץ התיעוד (swagger) להכיל הסעיפים הבאים, לכל הפחות, וכל סעיף נוסף שיכול לעזור למשתמש לצרוך את השירות באופן מיטבי:

- **תיאור השירות**: חשוב לכתוב את תיאור השירות באופן ברור ולא להיכנס להגדרות עסקיות מורכבות. יש להתייחס למי שקורא את השירות כאל משתמש שאינו בקיא במונחים העסקיים, ולהסביר באופן ברור את מטרת השירות.
- **חוקיות עסקית**: באילו תרחישים המידע זמין/באילו תרחישים המידע לא זמין.
- **תרחישי שגיאות**, כולל תיעוד של קודי שגיאה והודעות אפשריים.
- **תיעוד על שירות** שזמין לצורך בדיקת השירות, לרבות הסבר למידע פיקטיבי שניתן לשלוח לצורך הפקת תגובה.
- **תיעוד על אופן הפקת שגיאה** באופן יזום לצורך בדיקת תרחיש של שגיאה.
- **הסבר מלא על אופן הפניה** (request) **והתגובה הצפוי** (response). ככל שיש יותר מאפשרות אחת, יש לציין את כל האפשרויות.
- **פירוט מלא של המידע שחוזר** כולל הסבר מפורט לכל שדה והערכים שיכולים לחזור.
- מידע על הרשאות גישה לשירות.
- מידע על שינויים עתידיים צפויים.
- מידע על אופן ניהול גרסאות השירות.
- מידע על גרסאות קודמות לשירות.
- האם יש מגבלה למספר הפניות שניתן לבצע בזמן נתון (rate limit).
- האם יש מגבלה לשעות שהשירות זמין.
- שירות שנדרש לשלם עבורו – יש לציין את ההיבט הכספי של השירות.

7.3.3 סוגי נתונים Data types

יש להיצמד לסוגי הנתונים כפי שמוגדר ב- OpenAPI 3 Specification Data Types.

7.3.4 אימות Authentication

- <u>שירותים המתפרסמים בשדרת המידע (Outer API)</u>

תהליך הזיהוי יתבצע בין מוצרי ה - API gateway.

תהליך זה מתבצע בשתי שלבים:

1. שלב הפיתוח – בפורטל המפתחים (API service catalog). בשלב זה המפתח שמעוניין להשתמש בשירות שפורסם ומוגדר כ- private, יקבל מה -API Gateway ערך שיוגדר כ- Token. יש לשמור ערך זה לשלב הרצת השירות.
2. שלב הרצת השירות – ה-Token יוזן בפניה לשירות אשר יאשר את הזיהוי של צורך השירות. תזכורת, **אין לשמור ערך זה בקוד**. יש לנהל ערכים אלו במאגר חיצוני לקוד.

- <u>שירותים פנים ארגוניים (Inner API)</u>

במערכות פנים משרדיות יש להתייחס גם לתהליך הזיהוי הפנימי וההרשאות של משתמשים מזוהים של המשרד. בשלב ה-"כניסה" לשירות מתבצעת בדיקה שאכן יש הרשאות גישה לשירות. שלב הזיהוי הראשוני כולל את זיהוי ואימות המשתמש (תהליך ה- Authentication) אשר מתבצע מול Token שמועבר ממערכת ניהול ההרשאות החיצונית לשירות. לאחר שאושרה הכניסה לשירות יכולה להתבצע, בהתאם לצורך, בדיקה נוספת, בדיקה יישומית של הרשאות נוספות (תהליך Authorization), ובדיקת הרשאות אישיות ספציפיות לביצוע ברמת האפליקציה.

לדוגמא, כל עובדי משרד מורשים לצפות בנתונים ממאגר מסוים, אבל רק לדרגת הניהול הבכירה באותו משרד יש הרשאות עדכון. במקרה כזה, בהנחה שיש שירות אחד בארגון שפונה למאגר הנתונים, בדיקת רמת ההרשאות הפנימית תתבצע בתוך השירות, כאשר זיהוי העובד שנגיש לנתונים מתקיים מול מוצר חיצוני לניהול הרשאות.

שלבי הגישה:

1. הזדהות מול מוצר חיצוני לקבלת Token.
2. אישור כניסה לשירות.
3. בשירות תתבצע בדיקת הרשאות העובד שנכנס, למול הפעולות שמעוניין לבצע. רמת ההרשאה נמסרת מגישה למוצר לניהול הרשאות חיצונית למערכת. המערכת מקבלת את רמת ההרשאה ומאפשרת גישה/ פעילות לפי ההרשאה שהתקבלה.

7.3.5 מוסכמת שמות (Naming Convention)

"שם" הוא המזהה החזק ביותר שיש לכל רכיב תוכנה. שם טוב יסביר למשתמש בצורה ברורה מה מטרת השירות ולאיזה צורך הוא עונה. יש להקדיש חשיבה לשם שיינתן לשירות וכל המתודות שייחשפו באמצעות השירות.

אם שירות ייקרא citizen_Car_Status, במבט ראשון, כמשתמש, ניתן להבין שמדובר בשירות שייתן מידע על "סטטוס רכבים" ליישות של "אזרח".

לעומת זאת, אם ייקרא השירות cit_Car_Stat יש להניח שמשתמש שמעוניין להשתמש בשירות, לא יבין מיד במה מדובר, ואולי יחשוב שמדובר בשירות של כלי הרכב עבור עיר מסוימת.

כמו-כן, יש להתייחס לשירות כאל מוצר; וככזה - יש לשמור על עקביות השימוש בשם הראשי לאורך כל השימושים של השירות.

במשרד שמפרסם מספר שירותים, הצורך בעקביות במתן שמות, הינה קריטית למשתמשי השירות.

לדוגמא, משרד מסוים שמדבר על שירותים בעולם תוכן של ירושה, יקפיד להשתמש באותם מינוחים בכל השירותים. הוא ישתמש במילה inheritance באופן קבוע: search_My_Inhertinace, Request_Inherintance_Order וכדומה, ולא יהיה שירות בנושא ירושה שייקרא לדוגמה, requestMyLeagcyPapers.

להלן דוגמה מאתר גוגל:

API Name	Example
Product Name	`Google Calendar API`
Service Name	`calendar.googleapis.com`
Package Name	`google.calendar.v3`
Interface Name	`google.calendar.v3.CalendarService`
Source Directory	`//google/calendar/v3`
API Name	`Calendar`

7.3.6 שגיאות

- יש להתייחס לשגיאה אחת לפחות בכל הגדרת סכמה לשירות.

  אין לפרסם שירות ללא טיפול במצב שגיאה.

- יש להבדיל בין שגיאות מסוגים שונים:
  - שגיאות שנוצרו ממידע שגוי שהגיע מהבקשה (request).
  - שגיאות שנוצרו בשל בעיות ברשת או מקרים בקוד שלא טופלו.
- יש לפעול ככל הניתן לתפוס שגיאה ברמת הקוד ולא לשבש את תפקוד המערכת.
- לעולם אין לחשוף למשתמש (גם למשתמש פנים ארגוני וגם למשתמש חיצוני) את מהות השגיאות.

  חשיפת מידע על השגיאה יכולה להוות מפגע אבטחתי על ידי חשיפת פרטי מידע על הרשת. יש לתפוס את השגיאה במקום הכי קרוב להתרחשות השגיאה ולהעביר הודעה מתורגמת ללקוח.

  לדוגמא, אם יש שגיאה בפניה לבסיס נתונים בעקבות תקלת גישה ברשת, אין לשלוח את הודעת השגיאה שמתקבלת מהשרת, אלא יש להעביר שגיאה כגון "לצערינו התרחשה תקלה, נא לפנות שוב מאוחר יותר". הודעת שגיאה זו תועבר ללקוח ותוצג על גבי הממשק למשתמש הקצה. מידע מפורט על השגיאה ניתן לרשום למערכת פנים ארגונית לניהול שגיאות או למערכת לרישום ללוג ארגוני לצורך מעקב אחר התקלה והמשך טיפול.

  בעת רישום פרטי השגיאה למאגר פנימי, יש לוודא שיש רישום של פרטי מידע שיעזרו בשחזור וניתוח התקלה.

- למקרה של שגיאה בזמן ביצוע עדכון של מידע, יש לוודא שכל המידע הקודם משוחזר ולא מתאפשר מצב של עדכון חלקי של מידע.
- אין להחזיר הודעה HTTP response של "200 OK" במקרה של שגיאה.
- יש להשתמש ב- HTTP status codes. אלו הם קודים סטנדרטיים ומובנים היטב להעברת מידע בפניה לשירות API. ככל שניתן, יש להשתמש בשגיאה המתאימה ביותר.

  Hypertext Transfer Protocol [HTTP] Status Code Registry

- מומלץ לנהל את השגיאות באופן מרוכז ולפתח שירות תשתית ארגוני לטיפול בשגיאות. באחריות שירות זו יתבצע רישום ללוג, שליחת הודעת דוא"ל לגורם אחראי, רישום למאגר שגיאות, או כל פעולה אחרת ברמת הארגון כפי שהוגדר לטיפול במצבי שגיאה.

7.3.7 רישום ללוג

התעבורה שעוברת דרך API gateway נשמרת בלוג של ה- API Gateway.

רישום ותיעוד לוגים פרטניים המאפשרים ביצוע debug או מעקב אפליקטיבי על פניות יש לבצע בקוד למאגר נתונים יעודי.

ניתן, לדוגמה, לשמור במאגר נתונים פנים ארגוני, מידע על שגיאות שהתרחשו ולהחזיר לשירות הפונה מידע ממוסך (Masked error message) שכולל מספר שגיאה לשחזור השגיאה מתוך המאגר.

להלן מידע לדוגמא שניתן לשמור במאגר נתונים שמיועד ללוג:

- פרטי המערכת הפונה
- פרטי הפונה
- נתוני הפניה
- שעת הפניה
- מידע נוסף

7.3.8 גרסאות (Versioning)

בפרסום שירות לשימוש מול לקוחות, חשוב מאד לתמוך בגרסאות קודמות ולא "לשבור" את התמיכה בשירות. אחריות מפרסם השירות היא לדאוג לכך שהשירות יתמוך בשינויים לאחור.

שינויים אלו נחשבים שינויים שמחייבים שינוי מספר גרסה ופרסום השירות מחדש בפורטל השירותים:

- שינוי שם של API או פרמטרים קיימים ב- API.
- שינוי בהתנהגות השירות.
- שינוי בהודעות השגיאה המוחזרות מתוך השירות.
- שינוי בהתנהגות שעלולה להפתיע את המשתמש.

שינוי מספר גרסה מחייב התראה ללקוחות לפני ביצוע השינוי. ניתן ליישם התרעה ללקוחות באמצעות פורטל השירותים.

יש לפרסם גרסת שירות כגרסת "בטה" לפני הורדת השירות בגרסה הקיימת, על מנת לאפשר ללקוחות השירות להתנסות בה ולוודא שאכן השירות מגיב כצפוי.

7.3.9 הורדת שירות משימוש

אם שירות שנמצא בשימוש מיועד לרדת משימוש, יש להתריע על כך מראש, על מנת לאפשר ללקוחות השירות זמן היערכות.

אם יש שירות חדש שמחליף את השירות שיורד, יש ליידע על כך את הלקוחות.

יש לבחון, באמצעות כלי הניהול של שדרת המידע / API management שקיים בארגון, כמה לקוחות רשומים לשירות, על מנת להבין את מידת ההשפעה של הורדת השירות.

7.3.10 Filter and paginate data

במקרים בהם צפויה שליפה של מידע רב (כגון - איתור של כל העסקאות שבוצעו בתקופה מסוימת), יש לנהל את כמות הערכים שצפויים להיות מוחזרים לאחר שהבקשה עוברת לשרת. כמות מידע גדולה מדי עלולה לגרום לחסימת DDOS. במקרים בהם כמות המידע שצפויה לחזור מהשירות הינה גדולה, יש לתכנן את אופן החזרת המידע. לדוגמא, ניתן לנהל את הדפדוף לקבלת המידע במנות או להגביל את כמות המידע שיוחזר ללקוח. בכל מקרה מידע על אופן קבלת מידע רב יופיע בתיעוד השירות.

7.4 פיתוח מאובטח

7.4.1 הנחייה

קיימת הנחיית יה"ב 5.13 – "פיתוח מאובטח". ההנחיה נמצאת באתר יה"ב. אתר זה נגיש לממונה על הגנת הסייבר במשרד. יש ליישם הנחייה זו בשלבי הפיתוח.

7.4.2 ערכי התצורה

ערכי התצורה של פרויקט (ערכי key-value) יופרדו משכבת הקוד וינוהלו באופן נפרד.

דוגמא לערכים שמשתנים לפי סביבת יעד, הם כדלקמן:

- בפניה לבסיס נתונים, הערך של ה- connection string. ערך זה משתנה בפניה לבסיס נתונים שנמצא בסביבת הבדיקות, לעומת פניה לבסיס נתונים שנמצא בסביבת הייצור.
- בדומה ל-connection-string, בעת פניה ל- API יש צורך לפנות לסביבות המתאימות. לדוגמא, בפיתוח מערכת יש צורך לפנות ל- API בכתובת של הבדיקות, ואילו בסביבת ייצור הערך יהיה תואם לסביבת הייצור. במקרה שיש מספר סביבות ייצור, הערך יכול להשתנות בהתאם.

- אסור להכיל בתוך הקוד ובתוך קבצי התצורה נתונים חסויים, כגון מפתחות גישה או סיסמאות.

  ניתן להכיל מידע רגיש בכספת, במאגר נתונים, במוצר ה- pipeline.
- אין להכניס כתובת IP לתוך הקוד.

7.5 בדיקות שירות API

עולם הבדיקות הינו רחב ויש לו מתודולוגיה מסודרת משלו. במסמך זה לא יפורטו סוגי הבדיקות ואופן ביצוע הבדיקות, אבל תודגש חשיבות ביצוע בדיקות על שירותים, ובעיקר על שירותים שמתפרסמים בשדרת המידע לשימוש לקוחות מחוץ למשרד המפתח.

יש לבצע בדיקות מקיפות על השירותים, כגון - בדיקות שפיות, בדיקות פונקציונליות ובדיקות עומסים.

יש להקפיד שהבדיקות יכסו את כל התרחישים שהשירות חושף ולבצע בדיקות מסוגים שונים. להלן מספר דוגמאות של בדיקות שניתן לבצע:

- בדיקת תגובה תקינה והחזרת תשובה תקינה:
  - העברת מידע תקין עם תשובה (לדוגמא - שם יישוב לקוד ישוב קיים).
  - העברת מידע תקין ללא תשובה (לדוגמא - שליחת מספר תקין שאין לו יישוב משויך).
  - אם המידע שחוזר הינו רשימה, יש להתייחס לגורמים הבאים:
    - יכולת קבלת רשימה של מידע
    - רשימה ארוכה

- רשימה קצרה
- ללא תוצאות

- בדיקת תגובה תקינה במקרה של תוצאה שלילית. דוגמא למקרים הבאים שיש להתייחס אליהם:
  - בדיקת החזרת שגיאה ברורה במקרה של פניה עם נתונים לא תקינים מבחינת הלוגיקה העסקית שמוגדרת למערכת.
  - שגיאת "סכמה" על מבנה לא תקין:
    - העברת מידע לא תקין (לדוגמא - העברת מידע אלפאנומרי במקום מידע נומרי).
    - שליחת בקשה למידע שלא בפורמט תקין (לדוגמא - תאריך נשלח DD/MM/YY במקום DD\MM\YYYY).
  - שגיאת SSL אם השירות דורש הזדהות ולא התבצעה הזדהות לשירות.
  - בדיקת מבנה התגובה (מבנה ה- HTTP response).
  - בדיקת Status Code תקין שמוודא תקשורת מול השרת (לדוגמא HTTP response code = 200).
  - בדיקת פורמט תקין על פי הסכמה שהוגדרה:
    - בדיקה שהמידע מוחזר כאובייקט JSON בהתאם להנחיות אבטחת מידע
    - בדיקה שהאובייקט חזר תקין על פי אפיון, ע"י בדיקה של JSON Schema.
    - בדיקה שכל הפרמטרים והערכים שמוחזרים ב- Response תואמים את הערכים המצופים לפי הבדיקה הספציפית שבוצעה.
  - פניה לשירות למקרה שבסיס הנתונים לא זמין.

ניתן להשתמש בכלים שונים לבדיקת השירות, כגון Postman, Swagger UI ובו- Fiddler.

משרד שהתחיל להטמיע תהליכי אוטומציה, יכול להפעיל קוד אוטומציה, לדוגמא באמצעות כלי ה- Newman (כלי להרצת קוד של ה-postman בשורת command line).

יש לבצע בדיקות עומסים בהתאם לשימוש הצפוי בשירות.

7.6 פרסום ב- API GATEWAY

מוצר ה- API Gateway מיועד לנהל את השירותים בארגון בהיבטים שונים: ניהול הרישום לשירות, ניטור השימוש, הנפקת מפתח (API key) לשימוש, ניהול התעבורה ועוד. ה- API Gateway חושף פורטל לפרסום השירותים עבור צרכני השירות.

- שירותים לשימוש פנים משרדי (Inner API) יתפרסמו ב API Gateway פנים משרדי. בנוסף לתשתית זו, לצורך ניהול השירותים המשרדיים, יש משרדים שמנהלים מוצרי API gateway פנים ארגוני.
- שירותים לשימוש בין משרדי ממשלה (Outer API) יתפרסמו ב- API Gateway המשרדי ובהתאם להנחיות שדרת המידע הממשלתי.
- שירותים לשימוש גורמים מחוץ לממשלה (Outer API) יתפרסמו ב- API Gateway המרכזי בממשל זמין בהתאם להנחיות שדרת המידע הממשלתי.

7.6.1 מידע על השירות

בשלב פרסום השירות יש מספר פרמטרים שיש להתייחס אליהם:

כתובת ה URL בקובץ ה Swagger מוגדרת באותיות קטנות

קיים מידע על השירות:

- שם השירות
- תיאור השירות
- גרסת השירות
- דוגמאות שימוש
- Response אפשריים (2XX , 4XX , 5XX)

7.6.2 ניהול התעבורה הנכנסת (limits)

בשלב פרסום השירות, יש להגדיר את העומס הצפוי בפניה לשירות, נכון לנקודת זמן מוגדרת מראש, האם יש נקודות שיא שיש צורך להיערכות בתשתית.

יש להגדיר את אופן תגובת השירות במקרה שכמות הפניות עוברת את העומס שהוגדר, לדוגמא - מה השגיאה הצפויה.

יש להיערך גם ברמת תשתית הארגון למצב של עומס פניות. יש לוודא שהרשת יכולה לעמוד בעומס הצפוי ולעומס המאופשר על ידי ה -API gateway שמנהל את הפניות.

7.6.3 הרשאות גישה

בפרסום שירות לכלי API management, יש להגדיר מי מורשה לפנות לשירות, והאם השירות ציבורי (public) או פרטי (private).

צרכן שמעוניין לצרוך שירות שמוגדר private יידרש לפנות לאחראי השירות לרישום לשירות. בתהליך זה צרכן השירות יקבל API key וישתלב כחלק מהפניה לשירות בזמן ריצה כמזהה להרשאת פניה לשירות.

7.6.4 בדיקת תקינות השירות לאחר הפרסום (health check)

שירות שמיועד להעברת מידע בין משרדי ממשלה או בין ארגונים לממשלה יפורסם בשדרת המידע הממשלתי.

על מנת ששדרת המידע תוכל לדגום את השרותים, על כותב השירות לממש מתודת בדיקה בכלל השירותים. המתודה תקרא: health-check, והיא תחזיר תשובה כדלקמן - {"status": "success"}.

Action Name: health-check

HTTP response status code: 200

Response payload: {"status": "success"}

HTTP Method: Get

7.6.5 איתור שירותים בקטלוג Data Discovery

תהליך פיתוח שירות אינו מסתיים בפרסום השירות. אין ספק שמפרסם השירות מעוניין שלאחר פרסום השירות תהיה חשיפה מירבית לצרכנים. על מנת לאפשר לצרכני שירותים לאתר את השירות על המפרסם לפעול להגדרת מילות מפתח שיעזרו בתהליך האיתור בפורטל.

בתהליך פרסום השירות יש לוודא שבקובץ ה- Swagger הוגדרו מילות מפתח (tag) מגוונות שיאפשרו לבצע איתור של השירות.

לדוגמא, שירות שמיועד להחזיר פרטי רכב של אזרח, ניתן להזין ערכי מילות מפתח הבאים: רכב, רישיון רכב, בעלות רכב, רישוי וכו'.

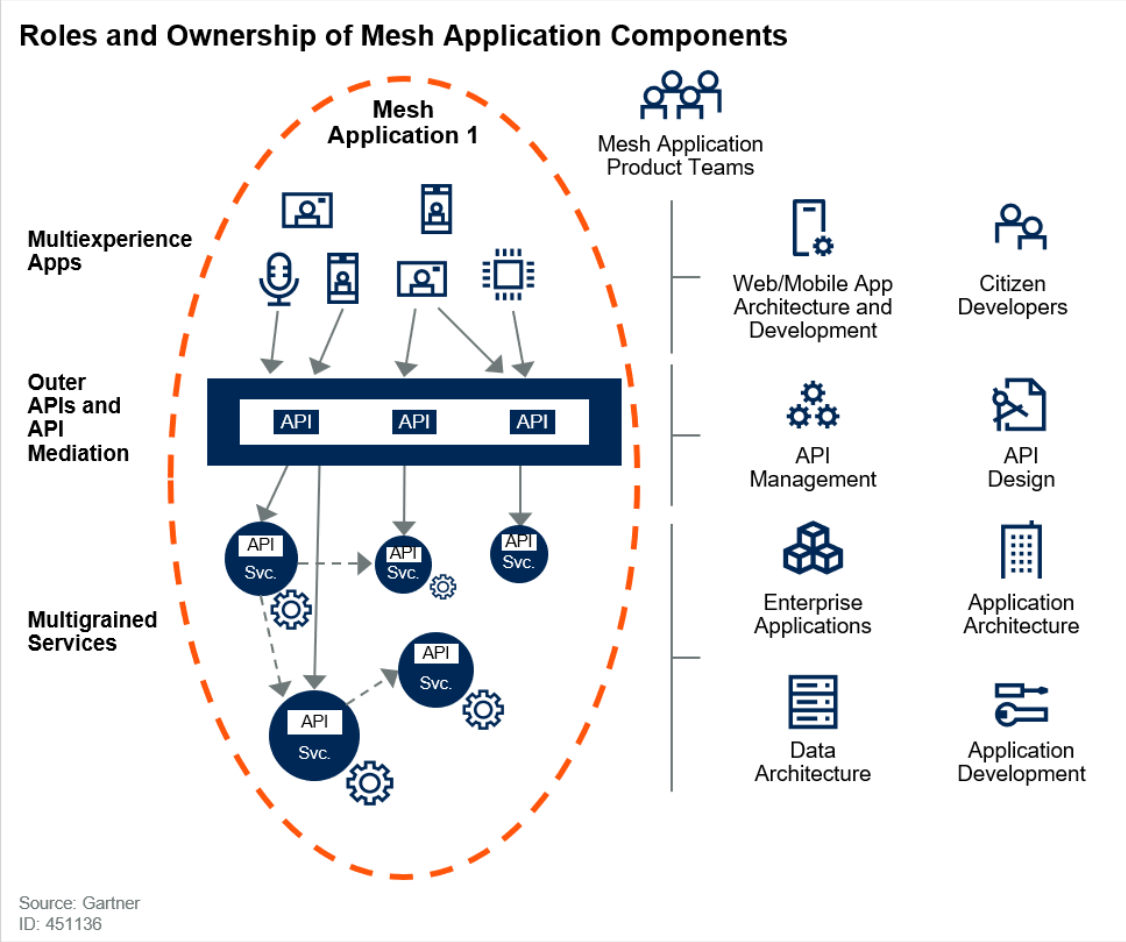
7.7 תמיכה, עדכון ותחזוקה

יש לראות בשירותי API מערכת טכנולוגית עבורה נדרש ניהול מחזור החיים. לצורך כך יש למנות גורם אחראי מתוך אגף טד"מ (בדומה למערכות המפותחות במשרד) שיהיה אמון על ניהול גרסאות והרחבה עתידיות, תחזוקת השירותים ומתן תמיכה שוטפת, תוך הקפדה על מענה ארוך-טווח לצרכי המשתמשים.

8. סיכום

עולם שירותי API הינו רחב. על מנת לנצל את הפוטנציאל הטמון בו על הממשלה להבין את היתרון של שיתוף מידע באמצעות שירותים, גם פנים ארגונית לייעול העבודה הפנימית, גם בין משרדי ממשלה וגם כלפי לקוחות מחוץ לממשלה. פרסום שירות אינו מסתיים בכתיבת שורות והעברת כתובת לצרכן השירות, אלא מתרחב להתייחסות אל שירות כאל מוצר שנדרש ניהול מחזור חיים מלא.

בתרשים ניתן לראות את השילוב של הארכיטקטורה של השירותים, כפי שהובא במסמך, לבין הרכיבים/גורמים השונים שמשתלבים בתהליך: ארכיטקטורת המערכת, לקוחות אזרחיים, ניהול השירותים, עיצוב השירות, מערכות ארגוניות, פיתוח מערכות וארכיטקטורת המידע.



9. קריאה נוספת

9.1 דיווח על אירועי סייבר ל SOC -הממשלתי, הנחייה 5.4

9.2 HYPERTEXT TRANSFER PROTOCOL (HTTP/1.1): SEMANTICS AND CONTENT

9.3 HYPERTEXT TRANSFER PROTOCOL (HTTP) STATUS CODE REGISTRY

9.4 עקרונות פיתוח מערכת להיערכות לענן

9.5 OPENAPI 3.0: HOW TO DESIGN AND DOCUMENT APIS WITH THE LATEST OPENAPI SPECIFICATION

10. גרסאות ההנחיה

מס'	סטאטוס	מהות שינוי	סעיפים שהושפעו	בתוקף מ-	נכתב ע"י	אושר ע"י
0.1	סופי	גרסה ראשונה		23.2.2021	קרן בר-לב	שחר ברכה