ChatGPT

Λεπτομερής Αρχιτεκτονική Automations για την εφαρμογή Astronote SMS Marketing

Αυτό το έγγραφο περιγράφει αναλυτικά τα βήματα (tasks) για κάθε αυτοματισμό που πρέπει να υλοποιηθεί στο Astronote – ένα πλήρες SMS marketing app που ενσωματώνεται με το Shopify. Ο σκοπός είναι να δώσει **όλη την απαραίτητη πληροφορία** ώστε ο κώδικας να υλοποιηθεί απρόσκοπτα από την ομάδα (ή από εργαλεία όπως το Cursor), αξιοποιώντας τις δυνατότητες της GraphQL Admin API, των webhooks, του Shopify Flow και του ήδη υπάρχοντος backend.

Γενικές αρχές υλοποίησης

1. **Αποθήκη δεδομένων και μοντέλα** – Δημιουργήστε collections/tables για *orders*, *fulfillments*, *abandonments* και *subscribers*. Αποθηκεύστε όλα τα δεδομένα που καταφθάνουν από τα webhooks (π.χ. JSON payloads), καθώς και τις επιλογές του χρήστη (banner opt-in, credits, templates). Αυτή η κεντρική αποθήκη θα επιτρέψει στο backend να ελέγχει αν μία αποστολή έχει ήδη γίνει, να χρεώνει credits και να προγραμματίζει jobs.

2. **Εγγραφή σε webhooks μέσω GraphQL** – Κάθε automation βασίζεται σε συγκεκριμένα webhooks. Χρησιμοποιήστε την mutation `webhookSubscriptionCreate` κατά την εγκατάσταση/αυθεντικοποίηση της εφαρμογής για να γραφτείτε στα topics `ORDERS_CREATE`, `FULFILLMENTS_CREATE`, `CHECKOUTS_CREATE`, και ό,τι άλλο απαιτείται. Ορίστε το `callbackUrl` στο backend σας (π.χ. `/webhooks/order-created`). Κάθε webhook πρέπει να επαληθεύεται με HMAC header.

3. **GraphQL queries/mutations** – Αξιοποιήστε τις queries της Admin API για να ανακτήσετε λεπτομερή δεδομένα. Για παράδειγμα, η query `order(id: $id)` δίνει πρόσβαση σε πεδία όπως το όνομα της παραγγελίας, τα line items και τις πληροφορίες του πελάτη [1]. Οι queries `abandonedCheckouts` και `abandonmentByAbandonedCheckoutId` επιστρέφουν πληροφορίες για εγκαταλελειμμένα καλάθια με recovery URL και line items [2]. Να αιτείστε μόνο τα απαραίτητα πεδία για να κρατάτε μικρά τα payloads.

4. **Job queue & scheduler** – Για εργασίες με καθυστέρηση (π.χ. 30 ή 60 λεπτά) ή επαναλαμβανόμενα jobs (win-back, restock), χρησιμοποιήστε έναν job queue όπως Bull (Redis), RabbitMQ ή οποιοδήποτε worker system υπάρχει ήδη στο project. Καθένα από τα παρακάτω tasks περιγράφει πότε πρέπει να προγραμματίζεται ένα job.

5. **Templating και παραμετροποίηση** – Ο έμπορος μπορεί να γράψει το δικό του μήνυμα SMS. Πρέπει να υλοποιήσετε ένα UI στο frontend το οποίο εμφανίζει τη λίστα των διαθέσιμων *placeholder variables* (π.χ. `{{order.name}}`, `{{customer.firstName}}`, `{{lineItems}}`) και επιτρέπει στον χρήστη να τις εισάγει. Το backend, μετά την ανάκτηση των δεδομένων από το Shopify, αντικαθιστά αυτά τα placeholders με τις πραγματικές τιμές.

6. **Διαχείριση credits** – Κάθε SMS καταναλώνει 1 credit. Πριν την αποστολή ελέγχετε το διαθέσιμο υπόλοιπο και ενημερώνετε ανάλογα τον χρήστη. Καταγράφετε την χρήση σε αρχείο “SMSLog” (order ID, automation, timestamp, template, status).

Automation 1: Order Created (Παραγγελία τοποθετήθηκε)

Σκοπός: Να στέλνεται αυτόματα SMS επιβεβαίωσης/ευχαριστίας και πιθανό cross-sell όταν γίνεται νέα παραγγελία. Ο Shopify Flow trigger *Order created* ενεργοποιείται όταν ο πελάτης υποβάλει παραγγελία ή όταν ένα draft order σημειωθεί ως πληρωμένο [3].

Task 1.1: Εγγραφή webhook και επαλήθευση

1. Καλέστε τη mutation `webhookSubscriptionCreate` με `topic: ORDERS_CREATE` και `callbackUrl: https://<backend>/webhooks/order-created`. Φροντίστε να ζητάτε τα scopes `read_orders` και `read_customers` στην OAuth διαδικασία.
2. Στο backend υλοποιήστε το endpoint `/webhooks/order-created`. Επαληθεύστε το signature με το shared secret του app.
3. Εξάγετε `order_id` από το JSON payload. Αποθηκεύστε το payload στη DB για αναφορά.

Task 1.2: Ανάκτηση λεπτομερειών παραγγελίας

Χρησιμοποιήστε GraphQL query για να πάρετε λεπτομερή στοιχεία παραγγελίας:

```
query GetOrderDetails($id: ID!) {
  order(id: $id) {
    id
    name                 # π.χ. "#1001" – θα χρησιμοποιηθεί στο SMS
    processedAt          # ημερομηνία/ώρα επεξεργασίας της παραγγελίας
    phone                # τηλέφωνο για fallback αν δεν υπάρχει στο customer
[1]
    totalPriceSet {
      shopMoney { amount currencyCode }
    }
    customer {
      firstName          # μικρό όνομα του πελάτη [4]
      lastName           # επίθετο του πελάτη [5]
      displayName        # πλήρες όνομα ή fallback σε email/phone [6]
      email              # email για πιθανή επιβεβαίωση [7]
      defaultPhoneNumber { phoneNumber } # προτιμώμενο τηλέφωνο
      locale             # γλώσσα (π.χ. "el", "en") – για μελλοντικό
localization
    }
    lineItems(first: 50) {
      edges {
        node {
          title           # τίτλος προϊόντος
          quantity        # ποσότητα κάθε προϊόντος
        }
      }
    }
    discountCodes        # λίστα με discount codes που χρησιμοποιήθηκαν [8]
    shippingAddress {
      address1 city country
    }
```

```
    # Το πεδίο fulfillments περιέχει τις αποστολές της παραγγελίας και
περιλαμβάνει tracking αριθμούς όταν γίνει η αποστολή
  }
}
```

Τι δεδομένα αξιοποιούμε:

- `order.name` χρησιμοποιείται στην κεφαλίδα του SMS (π.χ. «Παραγγελία #1001»).
- `customer.firstName` και `customer.displayName` για εξατομίκευση ("Γεια σου Μαρία").
- `lineItems.edges` για να αναφέρουμε τα προϊόντα (τίτλος/ποσότητα) στο μήνυμα ή να δημιουργήσουμε προτάσεις cross-sell με βάση τις κατηγορίες.
- `totalPriceSet` για να αξιολογήσουμε αν ο πελάτης είναι υψηλής αξίας και να στείλουμε διαφορετικό μήνυμα.
- `discountCodes` για αναφορά στο μήνυμα («Έχετε ήδη χρησιμοποιήσει τον κωδικό ...»).
- `shippingAddress.city/country` για να στείλουμε πιο εξατομικευμένες πληροφορίες παράδοσης.

Task 1.3: Επεξεργασία και αποστολή SMS

1. Δημιουργήστε helper συνάρτηση που δέχεται το αντικείμενο `order` και αντικαθιστά placeholders από το template του εμπόρου. Για παράδειγμα:

```
function renderTemplate(template, order) {
  return template
    .replace(/{{order.name}}/g, order.name)
    .replace(/{{customer.firstName}}/g, order.customer?.firstName || '')
    .replace(/{{totalPrice}}/g, order.totalPriceSet.shopMoney.amount)
    .replace(/{{currency}}/g, order.totalPriceSet.shopMoney.currencyCode)
    .replace(/{{lineItems}}/g, order.lineItems.edges
      .map(edge => `${edge.node.title} x${edge.node.quantity}`)
      .join(', '));
}
```

1. Παρέχετε default template όπως: «Ευχαριστούμε για την παραγγελία **{{order.name}}**! {{customer.firstName}}, μόλις παραγγείλατε: {{lineItems}}. Θα σας ενημερώσουμε μόλις σταλεί.» Ο έμπορος μπορεί να το αλλάξει από το UI.
2. Στο UI εμφανίστε λίστα διαθέσιμων variables: `{{order.name}}`, `{{customer.firstName}}`, `{{lineItems}}`, `{{totalPrice}}`, `{{shippingCity}}`. Αυτή η λίστα προέρχεται από τα fields του GraphQL payload.
3. Αφαιρέστε 1 credit από τον λογαριασμό του εμπόρου. Αν δεν υπάρχουν credits, αποθηκεύστε την αποστολή ως pending και ενημερώστε τον στην εφαρμογή.
4. Καλέστε την υπηρεσία SMS με το τελικό μήνυμα. Αποθηκεύστε response (status, messageId) στη DB.
5. **Προαιρετικό:** Αν το σύστημα υποστηρίζει, προγραμματίστε job 2–3 ημέρες μετά για upsell/cross-sell προτάσεις, αξιοποιώντας τα product tags. Π.χ. «Είδαμε ότι αγοράσατε {{lineItems}}. Δείτε αυτά που ταιριάζουν!».

Automation 2: Order Fulfilled (Παραγγελία εκπληρώθηκε)

Σκοπός: Να ενημερώνει τον πελάτη ότι η παραγγελία έχει αποσταλεί, να παρέχει tracking πληροφορίες και να ζητάει review/feedback.

Task 2.1: Εγγραφή webhook και επαλήθευση

1. Εγγραφείτε στο topic `FULFILLMENTS_CREATE` μέσω `webhookSubscriptionCreate`. Ζητήστε τα scopes `read_orders` και `read_fulfillments`.
2. Υλοποιήστε endpoint `/webhooks/fulfillment-created` που επαληθεύει το signature και αποθηκεύει το payload.
3. Το payload περιλαμβάνει `fulfillment_id` και `order_id`. Αποθηκεύστε τα σε DB με αρχική κατάσταση “pending_notification”.

Task 2.2: Ανάκτηση δεδομένων αποστολής

1. Χρησιμοποιήστε GraphQL query για να ανακτήσετε τόσο την παραγγελία όσο και το fulfillment:

```
query GetFulfillmentDetails($fulfillmentId: ID!) {
  fulfillment(id: $fulfillmentId) {
    id
    createdAt
    estimatedDeliveryAt   # εκτιμώμενη ημερομηνία παράδοσης [9]
    status                # π.χ. IN_TRANSIT, DELIVERED
    trackingInfo {
      number              # αριθμός αποστολής [10]
      url                 # σύνδεσμος παρακολούθησης [11]
      company             # όνομα courier (UPS, DHL κ.λπ.) [12]
    }
    order {
      id
      name
      customer { firstName lastName displayName phone }
    }
  }
}
```

1. Τα πεδία `trackingInfo.number` και `trackingInfo.url` σάς επιτρέπουν να συμπεριλάβετε link παρακολούθησης στο SMS. Η Shopify αναφέρει ότι το tracking URL προβάλλεται στους πελάτες και στην admin όταν παρέχεται είτε το πεδίο `url` είτε ένα γνωστό `company` και `number` [13].

Task 2.3: Αποστολή ειδοποίησης αποστολής και προγραμματισμός review

1. Δημιουργήστε template: «Η παραγγελία **{{order.name}}** έχει αποσταλεί! Παρακολούθηση: {{trackingUrl}} (αριθμός: {{trackingNumber}}). Σε ευχαριστούμε, {{customer.firstName}}.»
2. Αντικαταστήστε placeholders με τα πεδία που επιστρέφει το GraphQL: `trackingInfo.url`, `trackingInfo.number`, `customer.firstName`.
3. Χρεώστε 1 credit και στείλτε SMS μέσω provider.

4. **Scheduled review:** δημιουργήστε job που τρέχει 5–7 ημέρες μετά το `estimatedDeliveryAt` ή, αν λείπει, 5 ημέρες μετά το `createdAt`. Το job θα στείλει άλλο SMS: «Ελπίζουμε να απολαμβάνετε το προϊόν σας! Αφήστε μία κριτική στο σύνδεσμο: {{review_link}}». Χρησιμοποιήστε dynamic link αν έχετε review page.
5. Ενημερώστε τη βάση ότι στάλθηκε το SMS αποστολής και το SMS review.

Automation 3: Customer Abandons Checkout (Εγκατάλειψη καλαθιού/checkout)

Σκοπός: Να ανακτήσετε εγκαταλελειμμένα καλάθια στέλνοντας SMS υπενθύμισης με link επαναφοράς. Ο trigger **Customer abandons checkout** στο Shopify Flow ξεκινά όταν ο πελάτης εγκαταλείψει το checkout [14].

Επιλογή 1: Υλοποίηση με Shopify Flow

1. **Δημιουργία Flow:** Στο Flow UI, ορίστε νέο workflow:
2. Trigger: "Customer abandons checkout" [15].
3. Conditions: ελέγξτε τον αριθμό line items στην εγκαταλελειμμένη παραγγελία μέσω του αντικειμένου `Abandonment.abandonedCheckoutPayload.lineItems.edges.length`.
   - Αν `lineItems == 1`: branch A, wait 30 λεπτά.
   - Αν `lineItems >= 2`: branch B, wait 60 λεπτά.
   - Αν `lineItems == 0`: λήξη workflow.
4. Action: "Send HTTP Request" στο backend endpoint `/flow/abandoned-checkout` με body που περιέχει `abandonedCheckoutId`.
5. **Backend processing:** Το endpoint `/flow/abandoned-checkout` θα δεχτεί το `abandonedCheckoutId` και θα καλέσει την query `abandonmentByAbandonedCheckoutId` για να πάρει το αντικείμενο `Abandonment`.

```
query GetAbandonment($id: ID!) {
  abandonment(id: $id) {
    id
    abandonedCheckoutPayload {
      abandonedCheckoutUrl    # URL επαναφοράς για το checkout [16]
      lineItems(first: 10) {
        edges {
          node {
            title           # τίτλος προϊόντος [17]
            quantity        # ποσότητα κάθε προϊόντος [18]
            image { url }
          }
        }
      }
      subtotalPriceSet { shopMoney { amount currencyCode } }
      discountCodes        # κωδικοί έκπτωσης που χρησιμοποιήθηκαν, αν
υπάρχουν [19]
    }
    customer {
      firstName
      lastName
      phone
```

```
    }
    emailState            # κατάσταση email (SENT/NOT_SENT) [20]
    daysSinceLastAbandonmentEmail
    hoursSinceLastAbandonedCheckout【496914418189318†L212-L217】
  }
}
```

1. **Σύνθεση SMS:** Με βάση τα δεδομένα:
2. Αν `lineItems` έχει 1 προϊόν: Χρησιμοποιήστε μήνυμα τύπου «Γεια σου {{customer.firstName}}, ξέχασες το προϊόν **{{title}}** στο καλάθι σου. Ολοκλήρωσε την αγορά εδώ: {{abandonedCheckoutUrl}}».
3. Αν `lineItems` ≥ 2: «Έχεις {{count}} προϊόντα στο καλάθι σου ({{titles}}). Ολοκλήρωσε την αγορά εδώ: {{abandonedCheckoutUrl}}».
4. Μπορείτε να συμπεριλάβετε το `subtotalPrice` ή έκπτωση εάν ανιχνεύσετε κωδικό στο `discountCodes`.
5. **Αποστολή SMS και χρέωση credits:** Όπως στα προηγούμενα tasks.
6. **Δεύτερη / τρίτη υπενθύμιση:** Προαιρετικά, μπορείτε να προσθέσετε νέα wait steps στο Flow (24 ώρες μετά και 48 ώρες μετά). Ο οδηγός marketing προτείνει τρεις υπενθυμίσεις: η πρώτη 1 ώρα μετά, η δεύτερη μετά από 24 ώρες, και η τρίτη με κίνητρο όπως δωρεάν μεταφορικά [21].
7. **Τερματισμός workflow:** Αν στο μεταξύ ο πελάτης ολοκληρώσει παραγγελία (webhook `ORDERS_CREATE`), πρέπει να ακυρώσετε το scheduled job. Αποθηκεύστε mapping από abandoned checkout ID σε job IDs ώστε να μπορείτε να ακυρώσετε προγραμματισμένες αποστολές.

Επιλογή 2: Χωρίς Flow – Polling & Custom jobs

Αν δεν χρησιμοποιείτε Flow ή θέλετε πλήρη έλεγχο, μπορείτε να υλοποιήσετε own logic:

1. Εγγραφείτε σε webhooks `CHECKOUTS_CREATE` και `ORDERS_CREATE`.
2. Όταν έρχεται `CHECKOUTS_CREATE`, αποθηκεύστε το `checkout_id`, την ώρα δημιουργίας και τα line items.
3. Δημιουργήστε job που εκτελείται κάθε π.χ. 15 λεπτά: ελέγχει για κάθε `checkout_id` αν υπάρχει αντίστοιχο order (μέσω DB ή `orders(first:1, query:"checkout_id:"…")`). Αν δεν υπάρχει και έχουν περάσει 30 ή 60 λεπτά, θεωρεί το checkout abandoned και στέλνει SMS.
4. Για να πάρετε recovery URL, καλέστε query `abandonedCheckouts(first:1, query:"id:<checkout_id>") { nodes { abandonedCheckoutUrl lineItems { edges { node { title quantity } } } } }`. Το αντικείμενο `AbandonedCheckout` παρέχει URL επαναφοράς και line items [2].
5. Συνεχίστε όπως στα προηγούμενα βήματα για SMS.

Automation 4: Welcome Series (Εγγραφή από το banner)

Σκοπός: Όταν ένας επισκέπτης συμπληρώνει το banner sign-up form (opt-in για SMS), να στέλνουμε σειρά από SMS που συστήνουν το brand και προσφέρουν κίνητρο για πρώτη αγορά. Το δεδομένο προέρχεται αποκλειστικά από το frontend μας, όχι από Shopify, επομένως δεν υπάρχει webhook.

Task 4.1: Αποθήκευση συνδρομητή

1. Όταν ο πελάτης συμπληρώνει το banner, συλλέγετε `phone`, `firstName` (προαιρετικά), και `optIn` μέσω της φόρμας. Αποθηκεύστε τα σε πίνακα `Subscribers` μαζί με timestamp εγγραφής και πηγή (Shopify store).
2. Προσθέστε flag `hasPurchased` (default `false`) που θα ενημερώνεται όταν ο συνδρομητής κάνει παραγγελία (μέσω webhook `ORDERS_CREATE`).
3. Ελέγξτε αν υπάρχει ήδη στην DB για να αποφύγετε διπλές εγγραφές.

Task 4.2: Welcome sequence

1. **SMS #1 – Άμεσο καλωσόρισμα**: Την στιγμή της εγγραφής στείλτε μήνυμα καλωσορίσματος: «Καλώς ήρθες στην [brand]! Χρησιμοποίησε τον κωδικό *WELCOME10* για 10% έκπτωση στην πρώτη σου αγορά.»
2. **SMS #2 – 2–3 ημέρες μετά**: Αν `hasPurchased == false`, προγραμματίστε job να στείλει μήνυμα με προτάσεις ή παρουσιάσεις προϊόντων. Π.χ. «Ανακάλυψε τα αγαπημένα μας προϊόντα εδώ: <link>. Μην ξεχάσεις τον κωδικό WELCOME10».
3. **SMS #3 – 7 ημέρες μετά**: Αν ακόμη δεν έχει γίνει αγορά, στείλτε υπενθύμιση: «Τελευταία ευκαιρία! Ο κωδικός WELCOME10 λήγει σύντομα.».
4. Τα templates θα είναι επεξεργάσιμα μέσω UI – διατηρήστε placeholder για `firstName`.

Automation 5: Post-Purchase Series

Σκοπός: Να συνεχίζει την επικοινωνία μετά την αγορά, αυξάνοντας την ικανοποίηση και τις επαναληπτικές αγορές. Οι οδηγίες marketing προτείνουν email/SMS follow-ups όπως thank-you, ενημέρωση αποστολής, ζητήστε review, cross-sell κ.ά. [22] .

Task 5.1: Ενεργοποίηση

1. Η Post-Purchase sequence ξεκινά από το `ORDERS_CREATE` ή από το fulfillment job. Χρησιμοποιήστε ίδια δεδομένα όπως στο **Automation 1** για να εξατομικεύσετε τα μηνύματα.

Task 5.2: Σειρά μηνυμάτων

1. **SMS #1 – Ευχαριστήριο**: Αμέσως μετά την παραγγελία στείλτε μήνυμα: «Σας ευχαριστούμε για την αγορά σας, {{customer.firstName}}! Είμαστε εδώ για οτιδήποτε χρειαστείτε».
2. **SMS #2 – Μετά την παράδοση**: 5–7 ημέρες μετά το fulfillment, στείλτε μήνυμα: «Ελπίζουμε να απολαμβάνετε το {{lineItems}} σας! Θα χαρούμε να ακούσουμε τα σχόλιά σας – αφήστε κριτική εδώ: {{reviewLink}}».
3. **SMS #3 – Loyalty/Referral**: 10–14 ημέρες μετά, ενημερώστε για πρόγραμμα επιβράβευσης ή referral: «Κερδίστε πόντους με κάθε αγορά και μοιραστείτε τον μοναδικό σου κωδικό referral για έκπτωση σε φίλους!».
4. **Restock Reminder** (προαιρετικό): Για αναλώσιμα προϊόντα, προγραμματίστε job ~30 ημέρες μετά: «Το {{product}} σου ίσως τελειώνει. Ξαναπαράγγειλε εδώ: <link>».
5. **Personalization variables:** Από το `order` αντικείμενο, χρησιμοποιήστε `customer.firstName`, `order.name`, `lineItems.titles`, `shippingAddress.city`, κ.λπ.

Automation 6: Win-Back Sequence

Σκοπός: Να επαναφέρει πελάτες που δεν έχουν αγοράσει για μεγάλο χρονικό διάστημα (π.χ. 90–180 ημέρες).

Task 6.1: Ανίχνευση ανενεργών πελατών

1. Δημιουργήστε scheduled job που εκτελείται μηνιαία. Το job αναζητά συνδρομητές στη DB οι οποίοι:
2. Έχουν `hasPurchased == true`.
3. Η τελευταία τους παραγγελία είναι παλαιότερη από 90 ημέρες.
4. Μπορείτε να το υπολογίσετε είτε από το `Subscribers` table (πεδίο `lastOrderAt`) είτε μέσω query στη Shopify Admin API (π.χ. `customer.orders` ή `orders` filtered by customer ID).

Task 6.2: Αποστολή win-back μηνυμάτων

1. Στείλτε SMS: «Μας λείψατε, {{customer.firstName}}! Δες τα νέα μας προϊόντα και πάρε 15% έκπτωση με τον κωδικό COMEBACK». Περιλαμβάνει σύνδεσμο στο κατάστημα.
2. Προγραμματίστε δεύτερο μήνυμα 7 ημέρες μετά με μεγαλύτερο κίνητρο αν δεν υπάρξει μετατροπή.
3. Καταγράψτε εάν ο πελάτης ανταποκρίθηκε (ολοκλήρωσε αγορά). Αν ναι, ενημερώστε `lastOrderAt` και ακυρώστε μελλοντικά win-back.

Automation 7: Cross-Sell & Upsell

Σκοπός: Να αυξήσει τη μέση αξία παραγγελίας προτείνοντας συμπληρωματικά ή premium προϊόντα.

Task 7.1: Ανάκτηση προτεινόμενων προϊόντων

1. Μετά το `ORDERS_CREATE` ή μετά την παράδοση, χρησιμοποιήστε την Shopify Recommendations API (αν διαθέσιμη) ή custom λογική για να βρείτε σχετικά προϊόντα:
2. Καλέστε query `productRecommendations(productId: $id) { id title handle image { url } }` για κάθε line item.
3. Εναλλακτικά, ανακτήστε products από την ίδια συλλογή ή με παρόμοια tags/metafields.
4. Αποθηκεύστε τις προτάσεις σε DB ώστε να υπάρχουν διαθέσιμες όταν το job στείλει SMS upsell.

Task 7.2: Αποστολή Cross-Sell/ Upsell SMS

1. Δημιουργήστε template: «Απολαύστε το νέο σας {{productTitle}}; Δείτε αυτά που ταιριάζουν τέλεια: {{recommendedProducts}}».
2. Το placeholder `{{recommendedProducts}}` θα αντικατασταθεί με λίστα τίτλων/links από τις προτάσεις.
3. Στείλτε το SMS 3–5 ημέρες μετά την παράδοση ώστε ο πελάτης να έχει ήδη εμπειρία χρήσης.
4. Παρακολουθήστε conversions: καταγράψτε αν ο πελάτης αγοράζει κάποιο από τα προτεινόμενα προϊόντα και υπολογίστε ROI.

Διάθεση variables στο UI

Είναι κρίσιμο ο έμπορος να γνωρίζει ποια δεδομένα μπορεί να χρησιμοποιήσει. Για κάθε automation, δημιουργήστε στον πίνακα παραμετροποίησης λίστα διαθέσιμων placeholders. Προτείνεται η ακόλουθη δομή JSON (per automation):

```
{
  "orderCreated": ["order.name", "customer.firstName", "customer.lastName",
"totalPrice", "currency", "lineItems", "discountCodes",
"shippingAddress.city"],
  "orderFulfilled": ["order.name", "customer.firstName", "tracking.number",
"tracking.url", "estimatedDeliveryAt"],
  "abandonedCheckout": ["customer.firstName", "lineItems",
"abandonedCheckoutUrl", "subtotalPrice", "discountCodes"],
  "welcome": ["subscriber.firstName"],
  "postPurchase": ["customer.firstName", "lineItems", "reviewLink"],
  "winBack": ["customer.firstName"],
  "crossSell": ["productTitle", "recommendedProducts" ]
}
```

Το frontend μπορεί να αναλύει αυτό το JSON και να εμφανίζει τα placeholders σε μορφή dropdown. Ο έμπορος μπορεί να τα εισάγει σε οποιοδήποτε σημείο του template. Κατά την αποστολή, ο renderer αντικαθιστά κάθε placeholder με την πραγματική τιμή.

Συμπέρασμα

Η παρούσα αρχιτεκτονική καλύπτει **end-to-end** τη ροή κάθε αυτοματισμού. Για κάθε trigger, καθορίζονται:

- Τα σωστά webhooks και τα απαραίτητα access scopes.
- Οι queries της GraphQL Admin API με τα συγκεκριμένα πεδία που θα χρειαστούν, συμπεριλαμβανομένων των URLs επαναφοράς checkout, των line items και των προσωπικών στοιχείων πελάτη [2] [1].
- Οι μεταβλητές που είναι διαθέσιμες στο UI, ώστε ο έμπορος να γράφει τα δικά του μηνύματα.
- Η διαδικασία αποστολής SMS, διαχείρισης credits και προγραμματισμού επόμενων εργασιών.

Ακολουθώντας αυτά τα tasks, η ομάδα υλοποίησης θα μπορεί να δημιουργήσει μια επεκτάσιμη, προσαρμόσιμη και marketing-έτοιμη εφαρμογή, που μεγιστοποιεί την απόδοση των καμπανιών SMS, ανακτά εγκαταλελειμμένα καλάθια και βελτιώνει τη σχέση με τους πελάτες.

---

[1] [7] [8] Order - GraphQL Admin
https://shopify.dev/docs/api/admin-graphql/latest/objects/Order

[2] [16] [19] AbandonedCheckout - GraphQL Admin
https://shopify.dev/docs/api/admin-graphql/latest/objects/AbandonedCheckout

[3] Shopify Help Center | Order created
https://help.shopify.com/en/manual/shopify-flow/reference/triggers/order-created

[4] [5] [6] Customer - GraphQL Admin
https://shopify.dev/docs/api/admin-graphql/latest/objects/Customer

[9] Fulfillment - GraphQL Admin
https://shopify.dev/docs/api/admin-graphql/latest/objects/fulfillment

[10] [11] [12] [13] FulfillmentTrackingInfo - GraphQL Admin
https://shopify.dev/docs/api/admin-graphql/latest/objects/fulfillmenttrackinginfo

14 15 Shopify Help Center | Customer abandons checkout
https://help.shopify.com/en/manual/shopify-flow/reference/triggers/customer-abandons-checkout

17 18 AbandonedCheckoutLineItem - GraphQL Admin
https://shopify.dev/docs/api/admin-graphql/latest/objects/AbandonedCheckoutLineItem

20 Abandonment - GraphQL Admin
https://shopify.dev/docs/api/admin-graphql/latest/objects/abandonment

21 Shopify email marketing automation: 5 Secret Sales Boosts
https://www.firstpier.com/resources/shopify-email-marketing-automation-guide

22 Shopify Post-Purchase Emails: Examples & Automation for 2025
https://www.retainful.com/blog/shopify-post-purchase-email