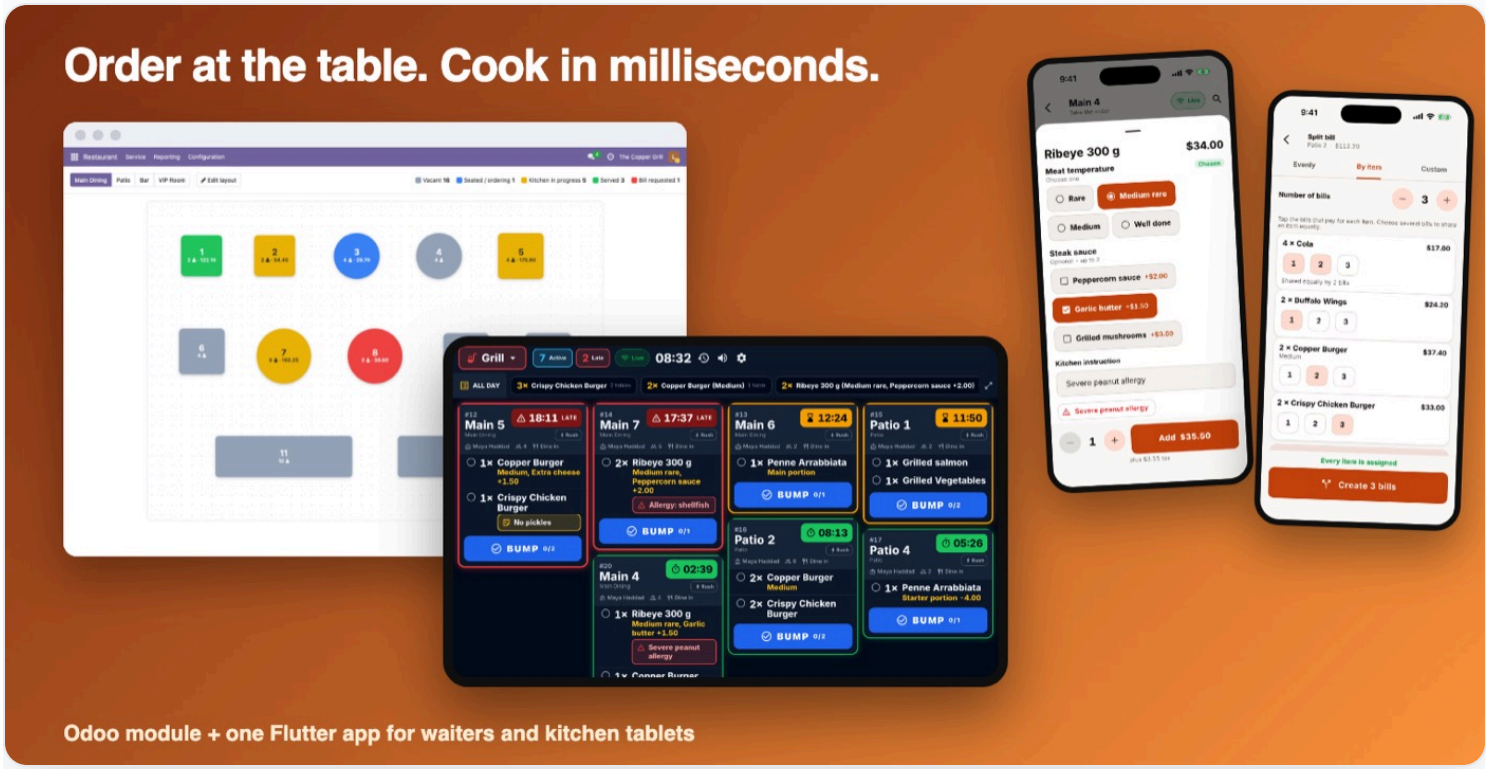
Odoo Restaurant Waiter App & KDS

A guide to the Odoo Restaurant Waiter App & Kitchen Display: installing the module, floors, menu, modifiers, stations, recipes, billing, the waiter app, the kitchen display, real time on the LAN, printing, reports and building the app.

1. Start here

In plain words: This package gives you a complete **table-side ordering and kitchen display system** for **Odoo Community**: an **Odoo module** (`taqnix_restaurant_kds`) that extends the restaurant point of sale, and the **Flutter source code** of **one app with two modes**: **Waiter** (phones and small tablets) and **Kitchen Display / KDS** (10 to 15 inch kitchen tablets). Waiters take orders at the table, the kitchen sees one ticket per station a fraction of a second later, and when the guests pay, Odoo books the sale in the open POS session and takes the ingredients out of the stock.



The Odoo floor map, the waiter app and the kitchen display.

> **Tip:** Nothing here needs Odoo Enterprise. The module uses only Community apps: Point of Sale with its restaurant extension, Manufacturing (for the recipes), Inventory and Invoicing. The live link between the devices is the websocket that every Odoo already has.

2. What is in the package

Item	What it is
`taqnix_restaurant_kds.zip`	The Odoo 19 module: floors and tables with live states, orders, courses with hold and fire, stations, tickets and SLA timers, modifiers, bills and splits, payments into the POS session, recipe consumption, reports, the API, Arabic translation and automated tests.
`flutter-app-source.zip`	The Flutter project (Android and iOS): the waiter mode and the KDS mode in one code base, offline outbox, real-time client with catch-up, ESC/POS printing over Bluetooth and over the network (port 9100), setup script, tests.
`documentation/`	This guide in English (`index.html`) and Arabic (`index-ar.html`), with PDFs, and `API.md` for developers.
`licensing/`	The licence (OPL-1: one purchase = one company and one app brand, updates included).
`preview-images/`	Screenshots of the apps and of the Odoo screens.
`changelog.txt`	History of the versions.

3. How a service works

Moment	Who	What happens	In Odoo
Seating	Waiter	Taps a grey table on the floor map and enters the number of guests.	An order opens; the table turns blue.
Ordering	Waiter	Picks dishes, chooses the forced options (meat temperature), the paid add-ons and writes the kitchen note. Appetizers and drinks are sent at once, mains and desserts stay *on hold*.	Lines with course, seat, options and prices from the menu.
Kitchen	Kitchen	A ticket appears on every station that has an item, with a chime. The timer runs green, orange, red. A tap on *Bump* says it is ready.	One ticket per station; the table turns yellow; ingredients are taken from the stock.
Pass	Waiter	Gets the notification, serves, and later *fires* the mains when the appetizers are cleared.	Lines become served; the table turns green.
Payment	Waiter	Asks for the bill, splits it, takes cash, card or a QR payment, adds the tip, prints the receipt.	The table turns red, then grey again; a POS order (and an invoice on request) is booked.
Back office	Manager	Watches the floor map, the sales by waiter, the preparation times and the ingredient use.	Reports and the normal POS accounting.

In plain words: Zero-latency on your own network. The devices talk to your Odoo server through the Odoo websocket. When Odoo runs on a small computer in the restaurant, the ticket is on the kitchen screen in a few milliseconds and nothing depends on the internet (chapter 21). If the Wi-Fi blinks, the waiter app keeps the order on the device and sends it when the link is back.

4. What you need

Item	Details
Odoo	19.0 Community or Enterprise. The apps **Point of Sale** with **Restaurant**, **Manufacturing**, **Inventory** and **Invoicing** are installed automatically with the module.
Chart of accounts	A chart of accounts with a cash journal and a bank journal (any localization). Taxes of the kind *percentage* are supported; tax-included prices work.
Server	For one restaurant: any small computer (4 GB RAM, SSD) at the restaurant, or a cloud server if you accept that the internet is part of the service. See chapter 21.
Network	A good Wi-Fi with all devices on the same network as the server. A fixed IP address for the server.
Devices	Waiters: Android 8+ phones or iPhones (iOS 15+). Kitchen: tablets of 10 to 15 inch, wall or stand mounted, always powered.
Printers	Optional: ESC/POS thermal printers of 58 or 80 mm, Bluetooth or on the network (TCP port 9100).
A computer to build the app	A Mac for iPhone and Android (Windows or Linux can build Android only), about 30 GB free, Flutter 3.41 or newer, Android Studio and Xcode.
Store accounts	Google Play Console and Apple Developer Program, or internal distribution to your own devices.

5. Install the Odoo module

1. Unzip the package. Inside is `taqnix_restaurant_kds.zip`; unzip it to get the folder `taqnix_restaurant_kds`.
2. Copy the folder into one of the `addons_path` folders of your Odoo server and restart Odoo.
3. Log in as an administrator, activate **Developer mode**, open **Apps** and press **Update Apps List**.
4. Remove the “Apps” filter, search **Restaurant Waiter App** and press **Install**.

```
# or from the command line
./odoo-bin -c odoo.conf -d YOUR_DATABASE -i taqnix_restaurant_kds --stop-after-init
```

> **In plain words:** After the installation a new app **Restaurant** appears in the main menu. The installation also creates the four courses (Beverage, Appetizer, Main and Dessert; the last two on hold by default) and a list of void reasons. Everything can be changed.

> **Important:** Take a backup before installing on a production database, as for any module. The dish products must be *not tracked in stock* (the ingredients are); otherwise the point of sale also takes the dish out of the stock when the session closes.

---

6. First setup in 20 minutes

1. Open **Restaurant → Configuration → Restaurants** and open your restaurant (a point of sale that is a restaurant). Tick *Waiter app and kitchen display*, add the payment methods and the floors.
2. Create the **floors and tables** (chapter 7) and arrange them in the floor map editor.
3. Put your dishes in the point of sale (*Available in POS*) with their menu category; create the **stations** and tell each station which categories it prepares (chapters 8 and 10).
4. Add **modifier groups** to the dishes that need them (chapter 9).
5. Give the dishes a **bill of materials** and choose the kitchen and bar **stock locations** (chapter 11).
6. Create the **users**: waiters, kitchen users with their station and a manager (chapter 13).
7. Open a POS session for the restaurant (the cashier or manager does it in Point of Sale → Dashboard), so payments can be booked.
8. Build the app with the setup script (chapter 24), install it on the devices, sign in and take a test order.

> **Tip:** The demo data of the screenshots (The Copper Grill) is created by `tests/seed.py` in the source package: it is a complete example of every setting.

---

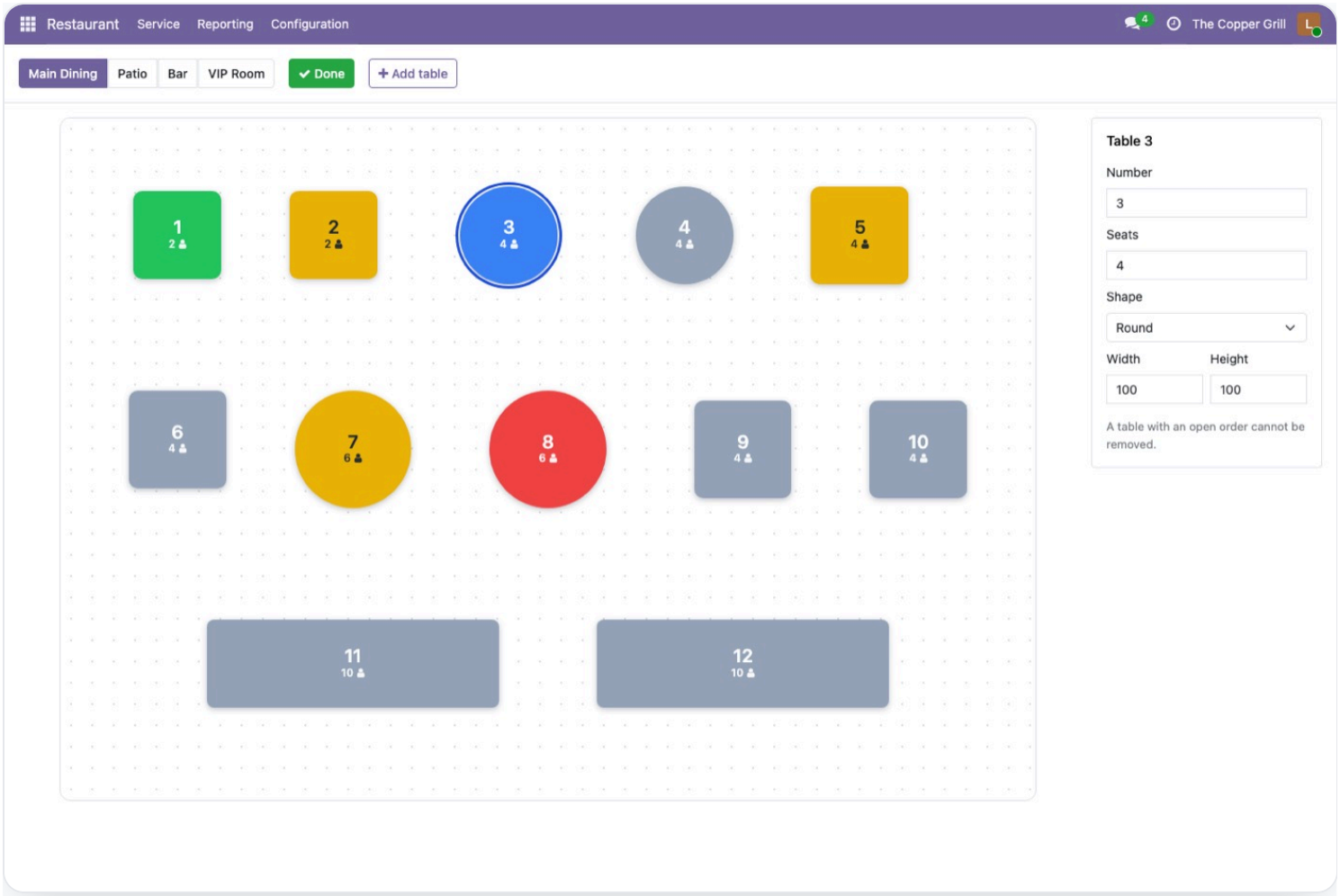
7. Floors, tables and the floor map

> **In plain words:** Floors and tables are the ones of the Odoo restaurant point of sale, so the standard POS screen shows the same plan. This module adds the zone of a floor (Dining, Patio, Bar, VIP), a short name for the apps (*Main 4*) and a third shape, the **long banquet**.

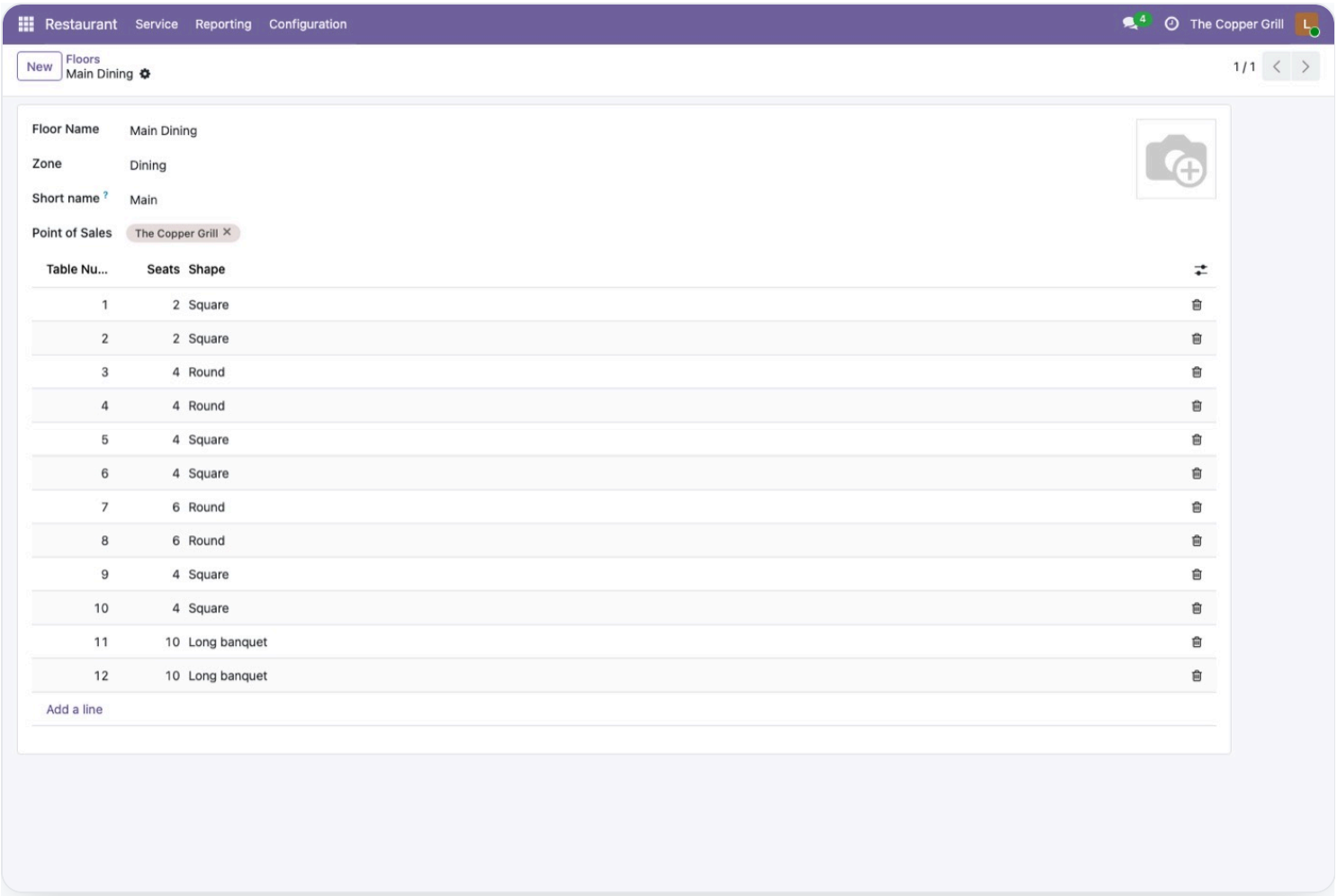
Restaurant → Service → Floor map: every table in its live colour, refreshed every few seconds.

Colour	State	Meaning
Grey	Vacant	No open order.
Blue	Seated / ordering	Guests are seated and the order is being taken, or only held items exist.
Yellow	Kitchen in progress	At least one item is in the kitchen or ready but not yet served.
Green	Served	Everything that was sent has been served.
Red	Bill requested	The bill was asked; the table is waiting for payment.

In plain words: **Layout editor.** Managers press *Edit layout* on the floor map: drag a table to move it (it snaps to a grid), click it to change the number, seats, shape and size, *Add table* for a new one and *Remove table* to archive one that has no open order. The waiter app shows the same plan at once.



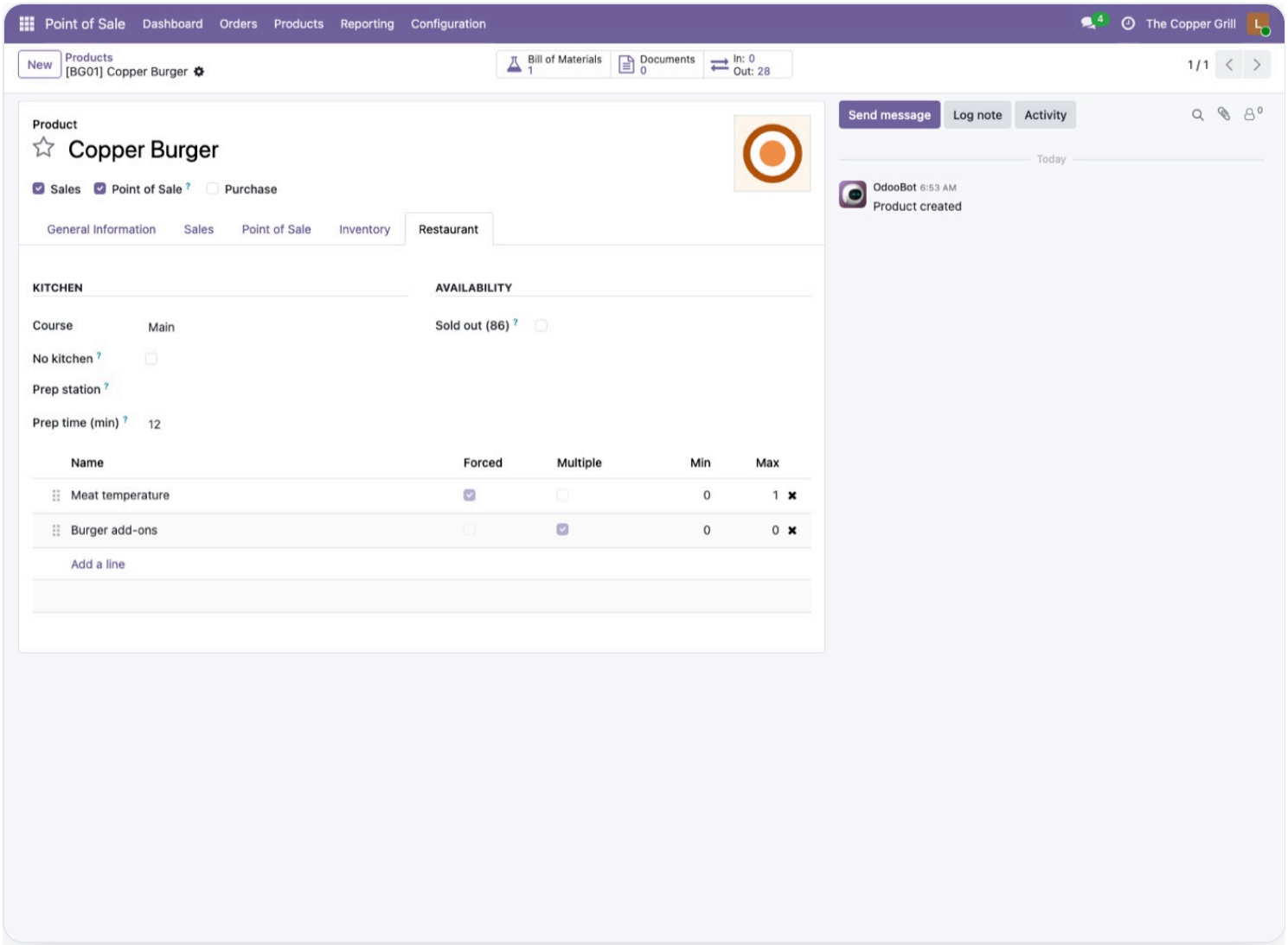
The layout editor: drag a table, click it to change its seats and shape.



A floor with its tables: capacity, shape and size.

8. The menu: categories, dishes and courses

> **In plain words:** The menu of the apps is the menu of the point of sale: products with *Available in POS* and their *POS categories*. Give each menu category an emoji in its form (*Icon*): the waiter app shows it on the category button.

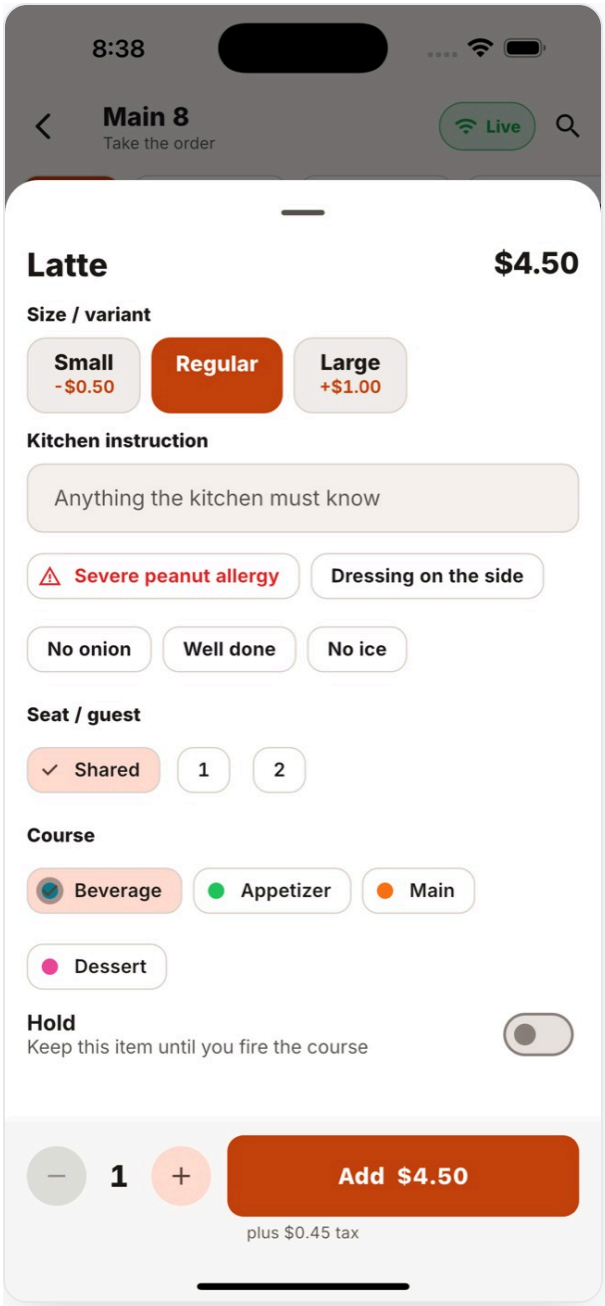


The Restaurant tab of a dish: course, station, preparation time, sold out and the modifier groups.

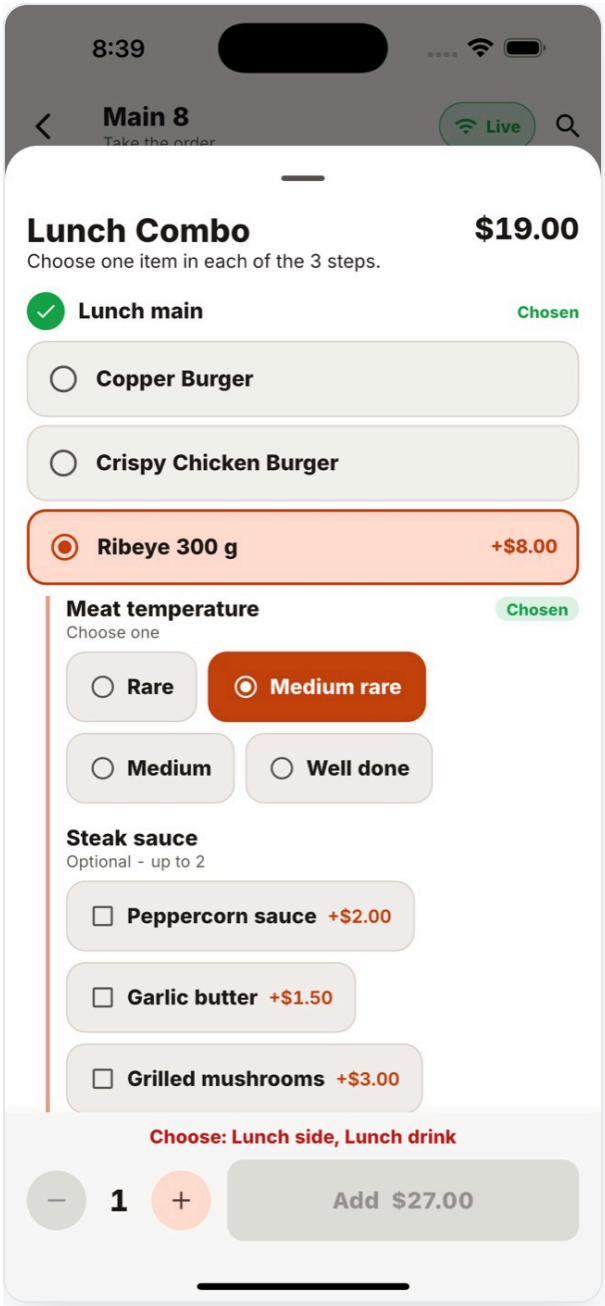
Field	What it does
Course	Beverage, Appetizer, Main, Dessert (or your own courses in Configuration → Courses). A course marked *Hold by default* keeps its items on hold until the waiter fires them.
Prep station	Overrides the routing by category: this dish always goes to that station.
No kitchen	The dish never goes to a screen and is ready at once (bottled water, packaged desserts).
Prep time (min)	Shown on the menu of the app as a hint for the waiter.
Sold out (86)	Waiters cannot add the dish until it is available again. Waiters and the kitchen can also switch it in the apps; every device is told at once.
Modifiers	The groups of options of the dish (chapter 9).

In plain words: Variants. Products with attributes (size, milk) appear as one card with a chip per variant; the price of each variant is the one set in Odoo (including the extra price of the attribute value).

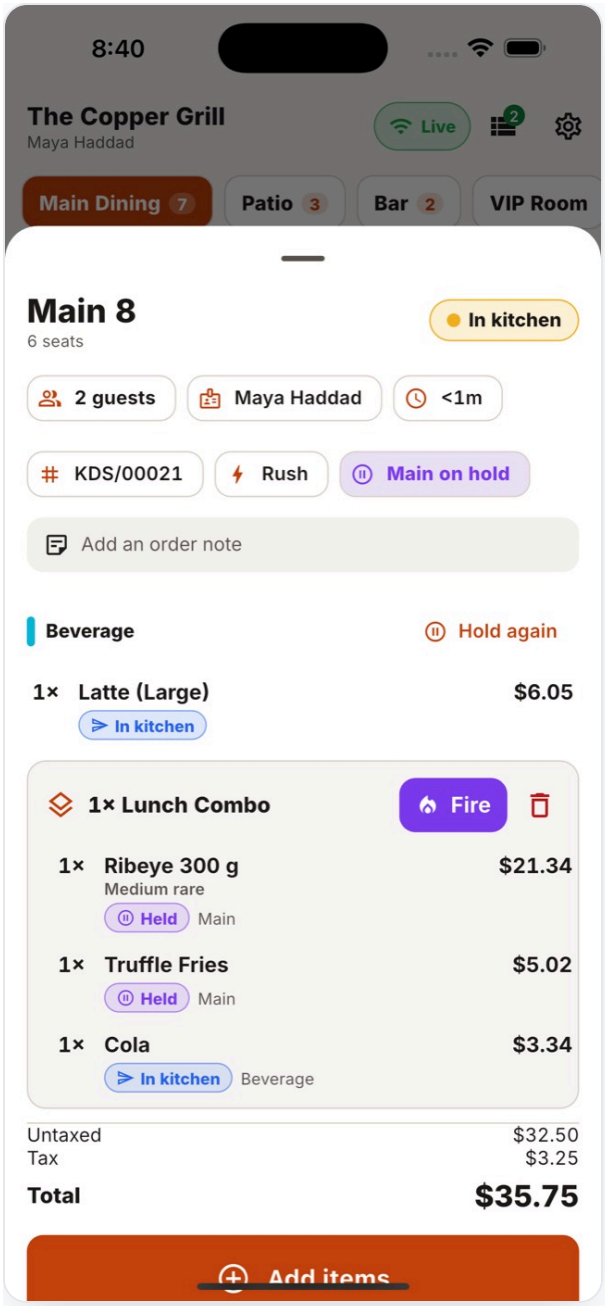
In plain words: Combos. A product of type *Combo* (Odoo's own combo feature: groups of choices with an optional extra price) is offered as one card. The waiter picks one item per group, and answers the forced options of each item. Odoo stores the combo as a header without price plus one line per item that carries its share of the combo price, so every item goes to its own station and course (the burger can wait on hold while the drink is served), bills and stock work per item, and voiding the combo voids all its items.



A variant product: one card, one chip per size.



A combo: one choice per group, with the options of each item.



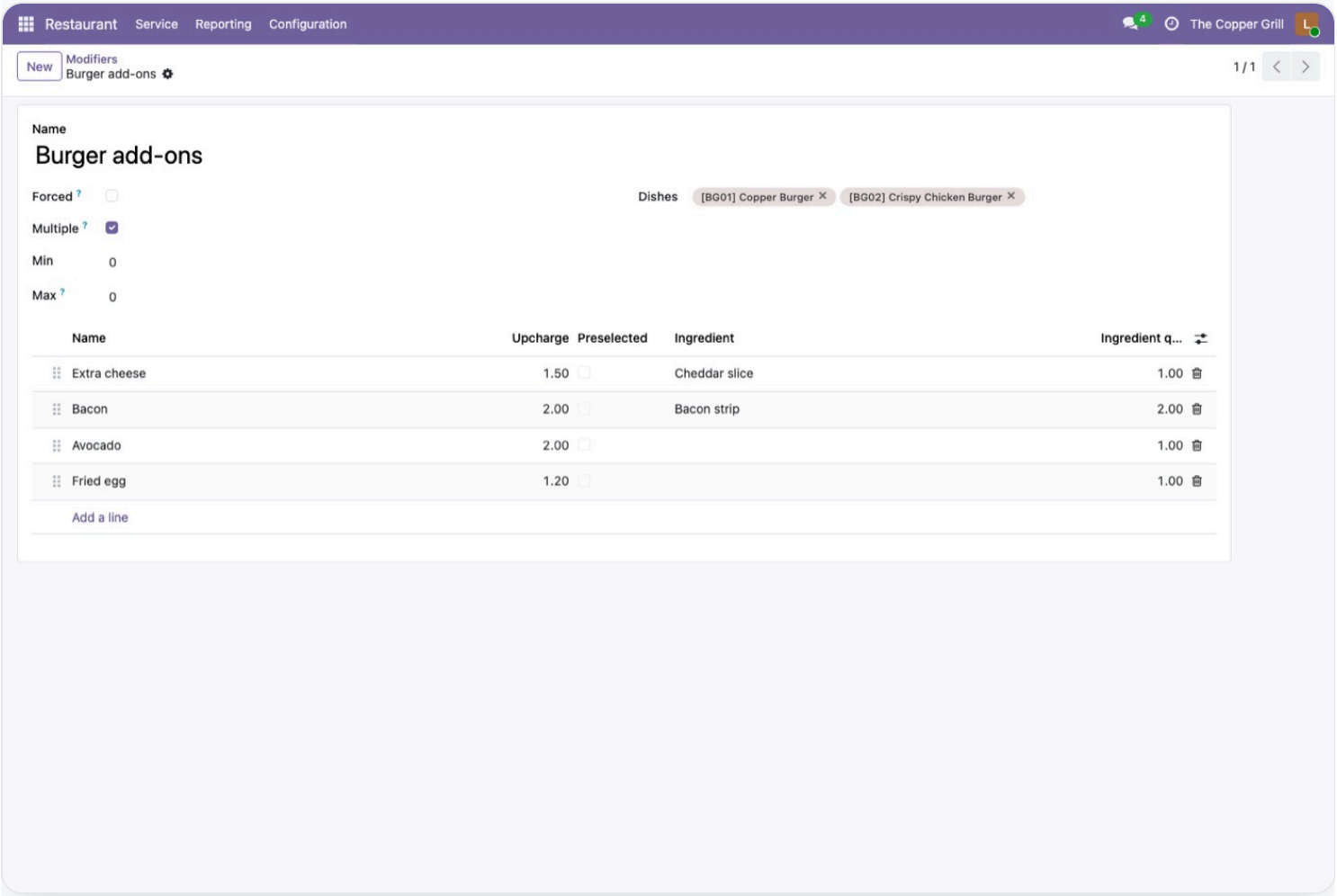
The combo in the order: the items are grouped under it.

> **Tip:** Prices come from the point of sale. If the restaurant uses a pricelist (for example a happy hour), the dish price follows it.

9. Modifiers: forced options, add-ons and notes

> **In plain words:** A **modifier group** is a set of choices. **Forced** groups (meat temperature: Rare, Medium rare, Well done) must be answered before the dish can be added. **Optional** groups carry

an **upcharge** (Extra cheese +1.50) that is added to the price and taxed like the dish. A group can allow one choice or several, with a minimum and a maximum.

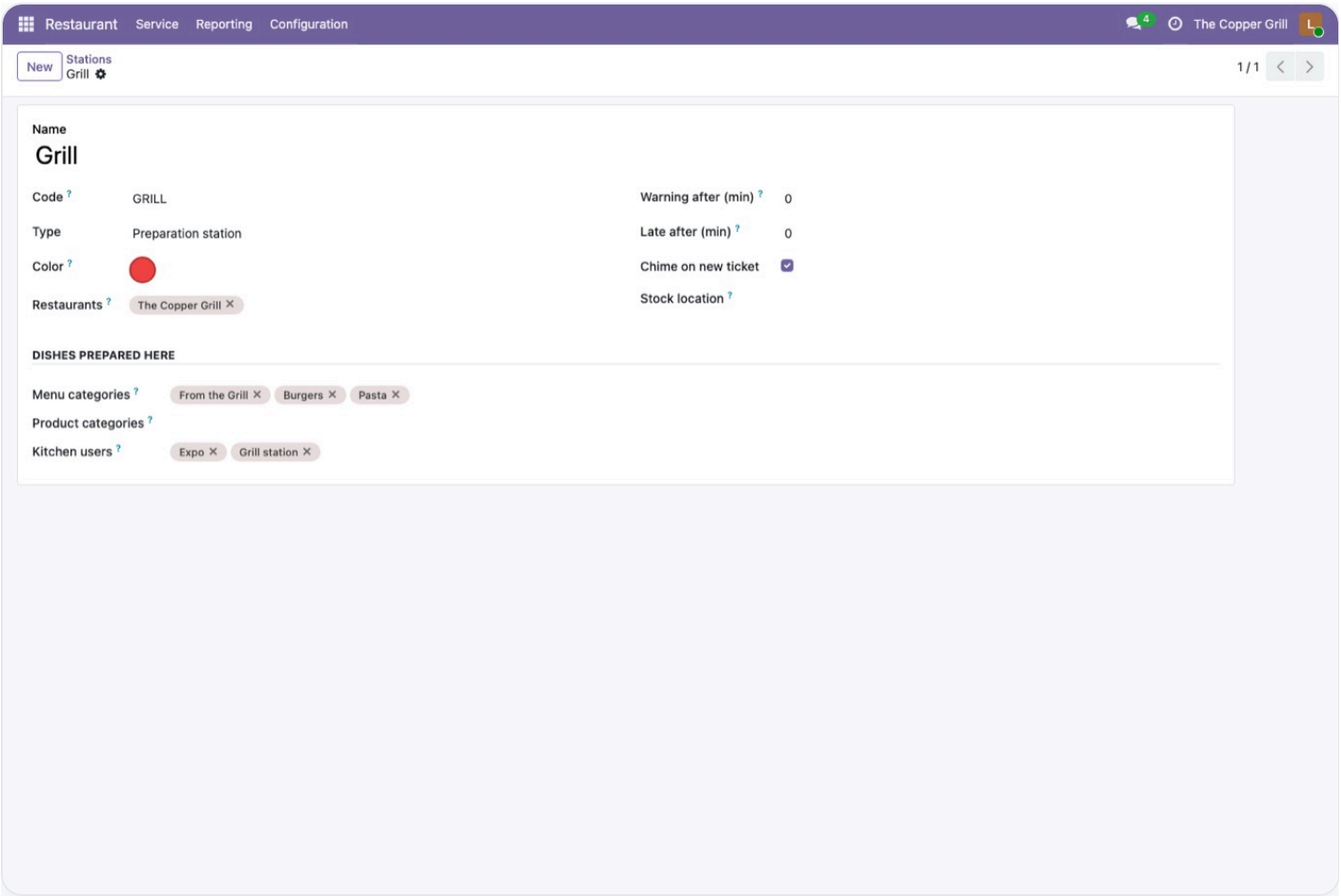


Burger add-ons: several choices, each with its price and, optionally, the stock ingredient it uses.

In plain words: An option can name an **ingredient** and a quantity: choosing *Extra cheese* then also takes one cheese slice out of the stock. The waiter also types a free **kitchen note** (for example *Severe peanut allergy*, *Dressing on the side*); the note is shown in a highlighted box on the ticket, and the app offers quick chips for the common ones.

10. Stations and smart routing

In plain words: A **station** is a screen: Grill, Fryer, Cold Prep, Bar. One order is **split into one ticket per station** automatically. The **Expo** station is special: it sees the tickets of every station.



A station: the menu categories it prepares, its stock location and its own timers.

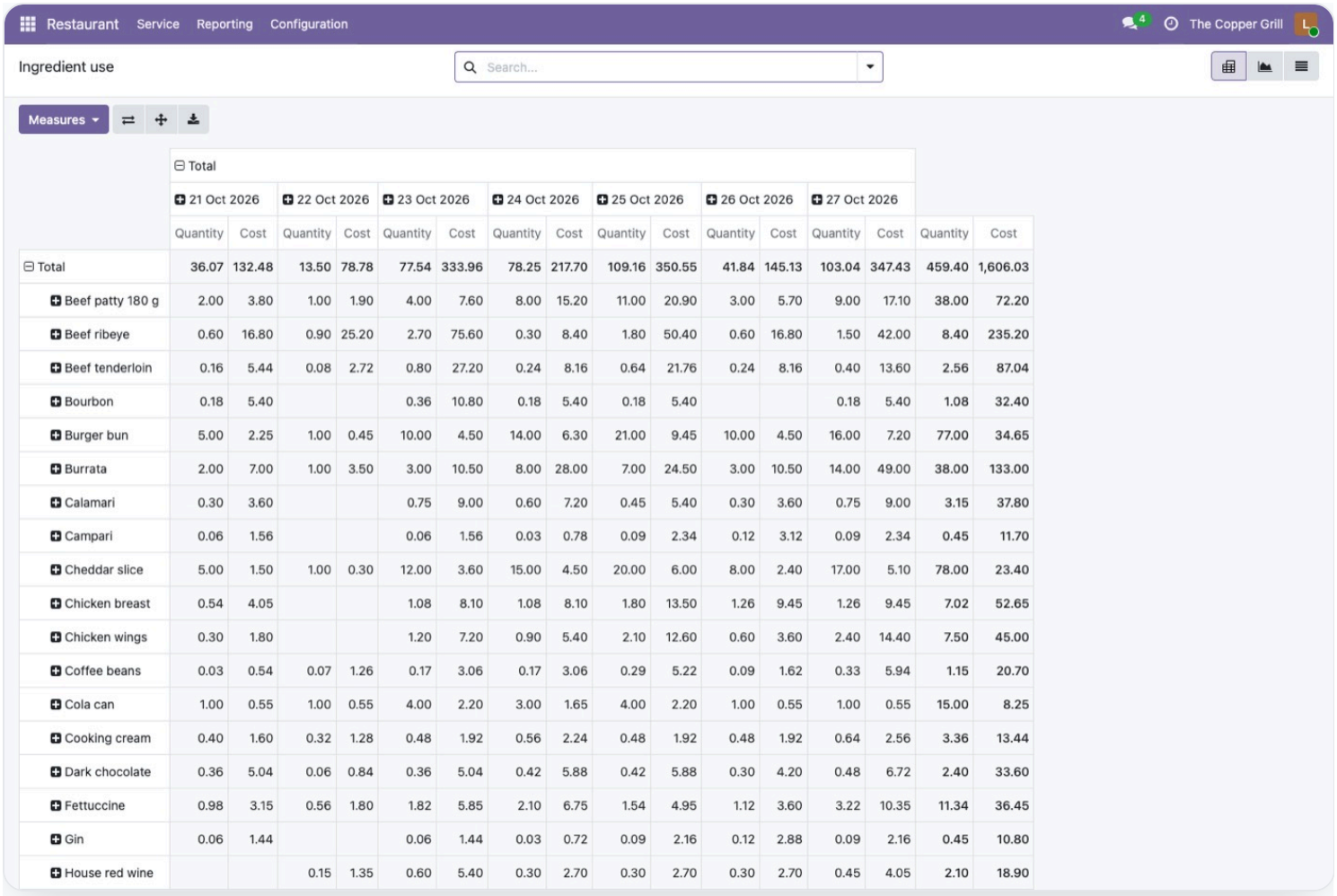
Rule (in this order)	Example
The dish names a prep station	Calamari → Fryer
A menu (POS) category or product category of the dish belongs to a station	From the Grill, Burgers, Pasta → Grill; Cocktails, Wine → Bar
The default station of the restaurant	Anything else → Grill
No station found and no default	The dish is not sent to a screen

In plain words: Timers. Each ticket counts from the moment it was sent: green until the warning time (10 minutes), orange until the late time (15), then flashing red. The times are set per restaurant and can be overridden per station (a salad bar is faster than a steak grill). The first time a ticket is late is stored for the reports.

11. Recipes and stock consumption

> **In plain words:** Give a dish a normal **bill of materials** in Inventory/Manufacturing and the module takes the ingredients out of the stock for you. Multi-level recipes work: a burger can use a *house sauce* that has its own bill of materials; with *All levels* the sauce is opened and mayonnaise and ketchup are consumed, with *Kit recipes only* the sauce itself is consumed.

Setting ***Consume ingredients***	**When the stock moves**
When the item is ready (bumped)	The moment the kitchen bumps it (default). A recalled ticket puts the ingredients back and takes them again at the next bump.
When it is sent to the kitchen	At the moment the ticket is created.
When the bill is paid	At the end, for everything that was served.
Never	No stock movement.



	Total															
	21 Oct 2026		22 Oct 2026		23 Oct 2026		24 Oct 2026		25 Oct 2026		26 Oct 2026		27 Oct 2026			
	Quantity	Cost	Quantity	Cost	Quantity	Cost	Quantity	Cost	Quantity	Cost	Quantity	Cost	Quantity	Cost	Quantity	Cost
Total	36.07	132.48	13.50	78.78	77.54	333.96	78.25	217.70	109.16	350.55	41.84	145.13	103.04	347.43	459.40	1,606.03
Beef patty 180 g	2.00	3.80	1.00	1.90	4.00	7.60	8.00	15.20	11.00	20.90	3.00	5.70	9.00	17.10	38.00	72.20
Beef ribeye	0.60	16.80	0.90	25.20	2.70	75.60	0.30	8.40	1.80	50.40	0.60	16.80	1.50	42.00	8.40	235.20
Beef tenderloin	0.16	5.44	0.08	2.72	0.80	27.20	0.24	8.16	0.64	21.76	0.24	8.16	0.40	13.60	2.56	87.04
Bourbon	0.18	5.40			0.36	10.80	0.18	5.40	0.18	5.40			0.18	5.40	1.08	32.40
Burger bun	5.00	2.25	1.00	0.45	10.00	4.50	14.00	6.30	21.00	9.45	10.00	4.50	16.00	7.20	77.00	34.65
Burrata	2.00	7.00	1.00	3.50	3.00	10.50	8.00	28.00	7.00	24.50	3.00	10.50	14.00	49.00	38.00	133.00
Calamari	0.30	3.60			0.75	9.00	0.60	7.20	0.45	5.40	0.30	3.60	0.75	9.00	3.15	37.80
Campari	0.06	1.56			0.06	1.56	0.03	0.78	0.09	2.34	0.12	3.12	0.09	2.34	0.45	11.70
Cheddar slice	5.00	1.50	1.00	0.30	12.00	3.60	15.00	4.50	20.00	6.00	8.00	2.40	17.00	5.10	78.00	23.40
Chicken breast	0.54	4.05			1.08	8.10	1.08	8.10	1.80	13.50	1.26	9.45	1.26	9.45	7.02	52.65
Chicken wings	0.30	1.80			1.20	7.20	0.90	5.40	2.10	12.60	0.60	3.60	2.40	14.40	7.50	45.00
Coffee beans	0.03	0.54	0.07	1.26	0.17	3.06	0.17	3.06	0.29	5.22	0.09	1.62	0.33	5.94	1.15	20.70
Cola can	1.00	0.55	1.00	0.55	4.00	2.20	3.00	1.65	4.00	2.20	1.00	0.55	1.00	0.55	15.00	8.25
Cooking cream	0.40	1.60	0.32	1.28	0.48	1.92	0.56	2.24	0.48	1.92	0.48	1.92	0.64	2.56	3.36	13.44
Dark chocolate	0.36	5.04	0.06	0.84	0.36	5.04	0.42	5.88	0.42	5.88	0.30	4.20	0.48	6.72	2.40	33.60
Fettuccine	0.98	3.15	0.56	1.80	1.82	5.85	2.10	6.75	1.54	4.95	1.12	3.60	3.22	10.35	11.34	36.45
Gin	0.06	1.44			0.06	1.44	0.03	0.72	0.09	2.16	0.12	2.88	0.09	2.16	0.45	10.80
House red wine			0.15	1.35	0.60	5.40	0.30	2.70	0.30	2.70	0.30	2.70	0.45	4.05	2.10	18.90

Restaurant → Reporting → Ingredient use: what was used, by dish, with its cost.

> **In plain words: Where from.** Each station can have its own stock location (the bar takes the liquor from the *Bar* location); the others use the *Kitchen stock location* of the restaurant. Stock moves go from there to the virtual production location.

> **In plain words: Waste.** An item that is voided after it was made keeps its ingredients consumed and is marked as waste, so the report shows what was lost.

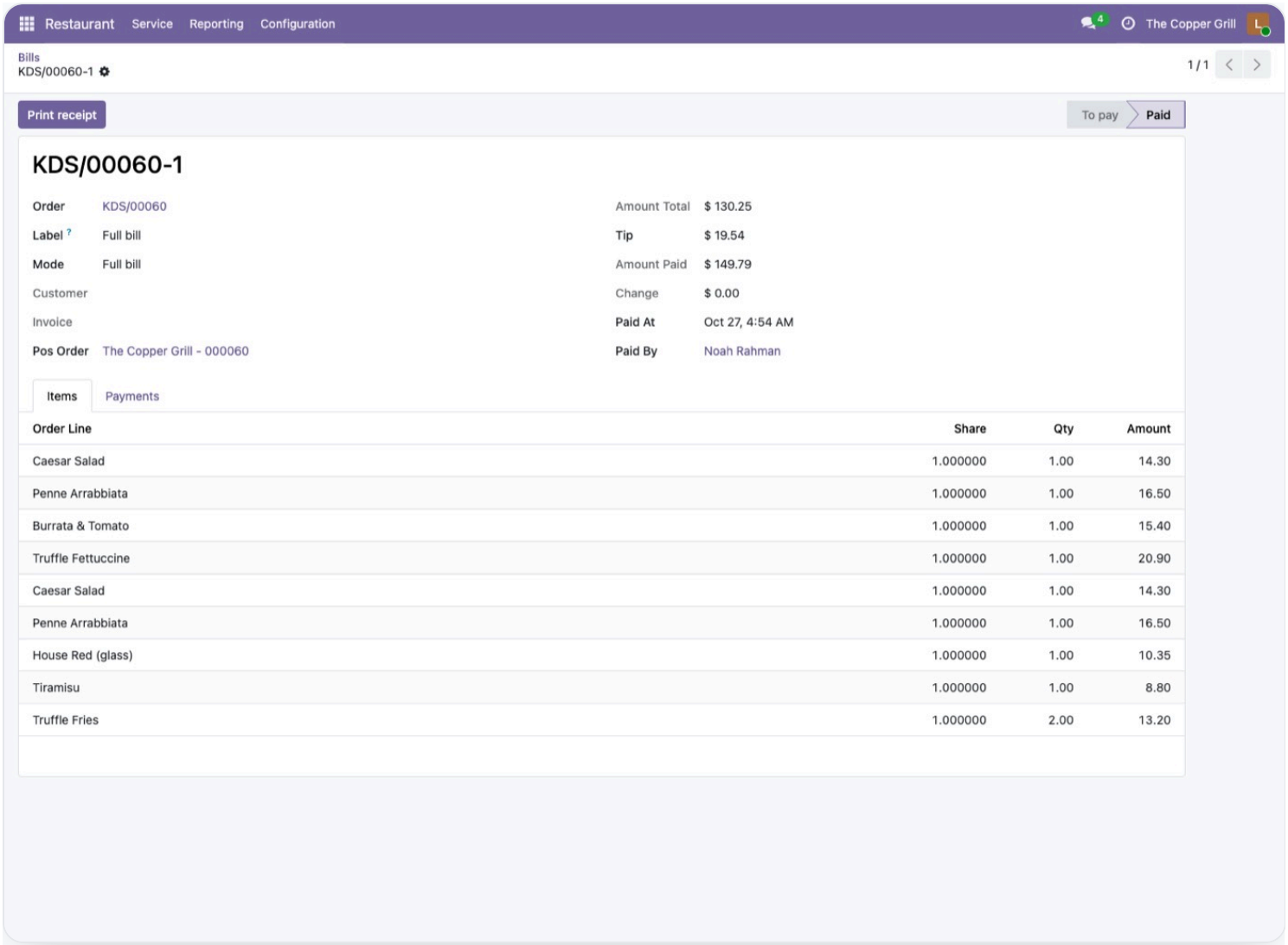
---

12. Bills, payments, tips and invoices

> **In plain words:** Paying works on **bills**. Asking for the bill creates one bill with everything on it; the waiter can split it. Every payment is collected in the apps; when the last bill is paid, the

module books the sale in the **open POS session** of the restaurant (a normal POS order with its payments, so your POS reports and closing work as usual).

Split	How it works	Booked in the POS as
None	One bill for the table.	One order.
Equal parts	Divide by the number of guests; the last guest takes the rounding cent.	One order with one payment per guest.
By item or seat	Each item (or a share of it, for a dish that the table shares) is given to a guest.	One order per guest.
By amount	Any amounts that add up to the total.	One order with the payments.



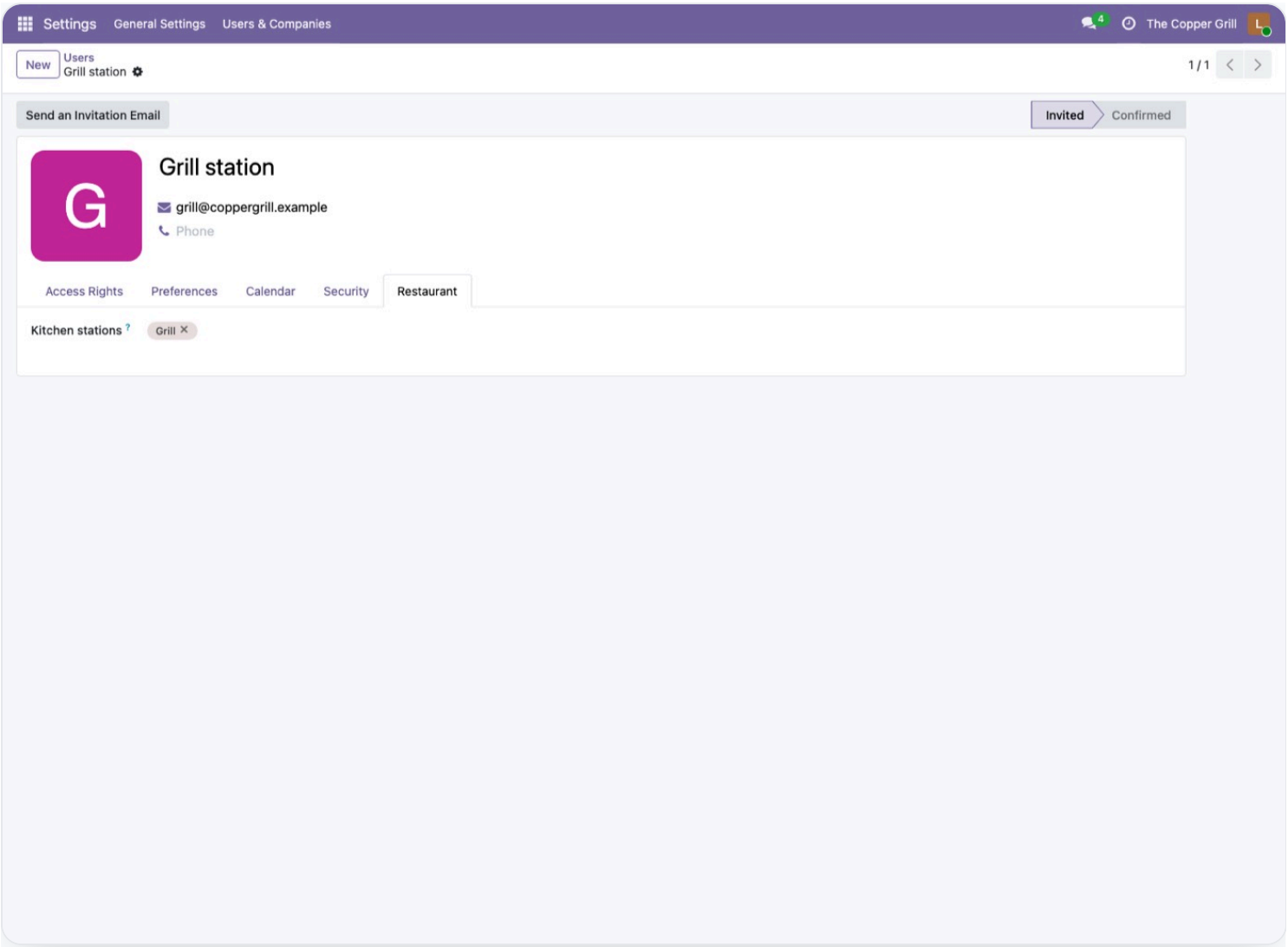
A paid bill in Odoo with its payments, the POS order and the invoice.

Topic	Details
Payment methods	The methods of the point of sale. Cash accepts more than the amount and shows the change. Mark a method *Show a QR code in the waiter app* to show the QR of the bill.
QR payment	In the restaurant form, *QR payment text* is the text of the QR with `{amount}`, `{currency}`, `{reference}` and `{company}`, for example a UPI link. Odoo cannot know that the guest paid by QR: the waiter confirms.
Tips	Enable *Product tips* in the restaurant form (and choose the tip product). The waiter picks a percentage or types an amount; the tip becomes a line of the POS order.
Invoice	At payment the waiter ticks *Invoice* and types the customer's name and tax number. Odoo creates or finds the customer and posts the invoice.
No open session	The app says *Open a point-of-sale session*. Open the session and press the button again.

> **Important:** Dishes must not be stored products: the point of sale takes sold products out of the stock again when the session closes. The ingredients are consumed by this module.

13. Users, roles and devices

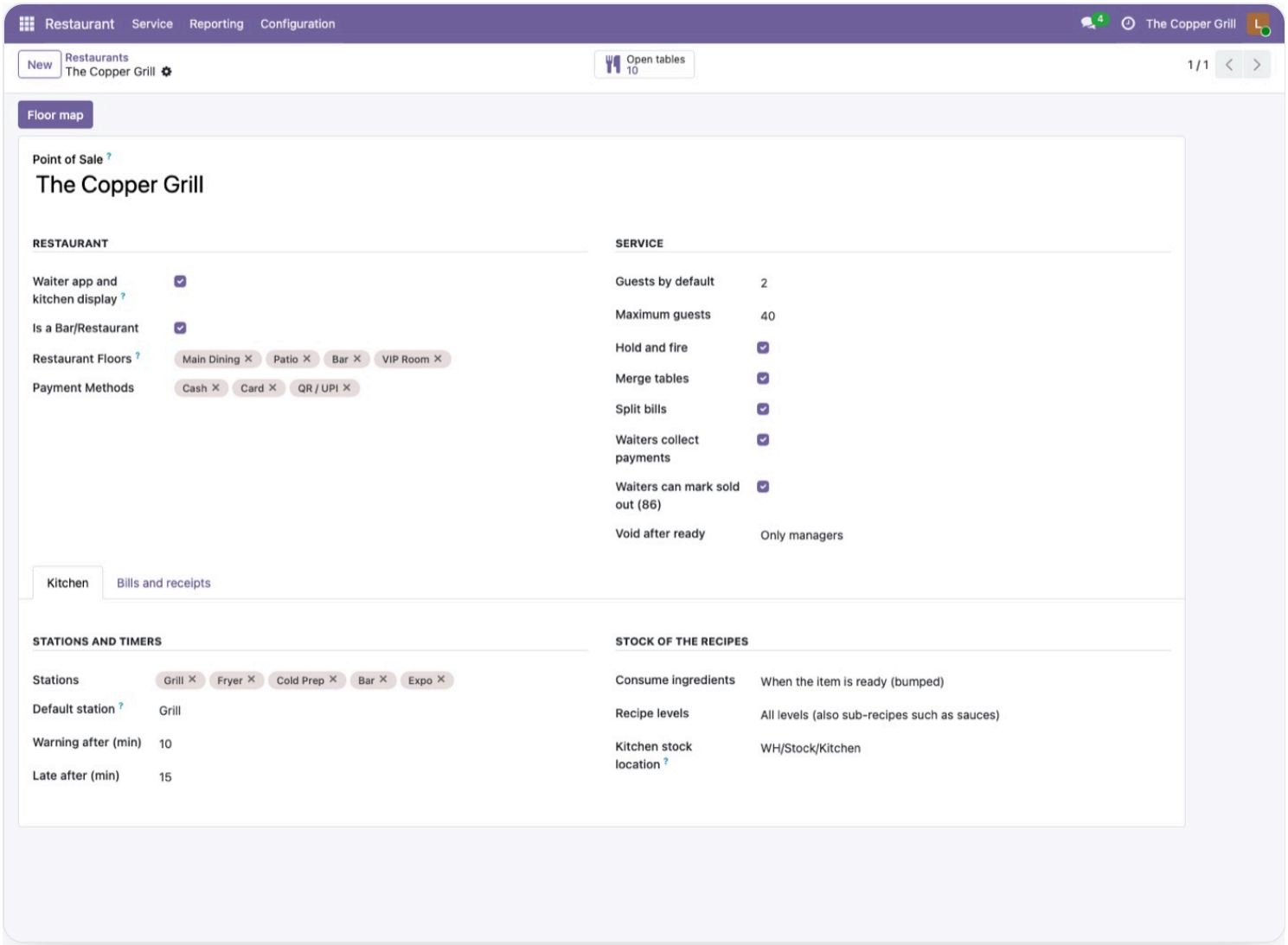
Group (Restaurant)	Can
Waiter	Sign in to the waiter mode, take orders, serve, ask and split the bill, collect payments (if allowed), mark an item sold out. In Odoo the waiter sees only their own orders.
Kitchen	Sign in to the KDS mode, see tickets, bump, recall and rush. Add stations in the user's *Restaurant* tab to limit the screens the user can open; empty means all.
Manager	Everything: voids prepared items, cancels orders, edits the setup, sees the reports. Implies Point of Sale manager rights.



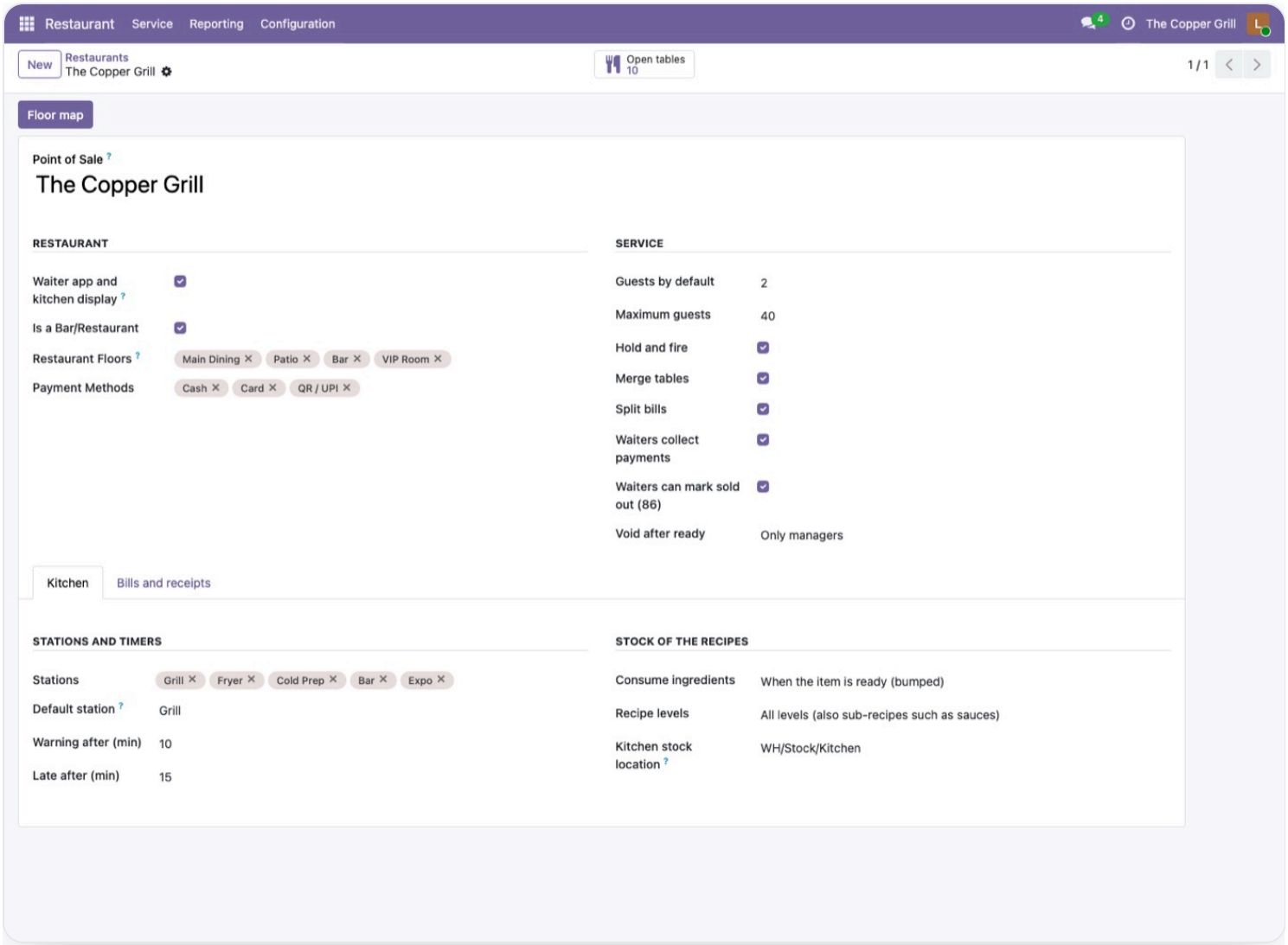
A kitchen user and the stations that user may open.

> **In plain words:** Every phone or tablet that signs in appears in **Configuration → Devices** with its app version and the last time it was seen. Delete a device to sign it out. Accounts with two-factor authentication cannot use the apps.

14. Restaurant settings reference



The restaurant form: service rules.

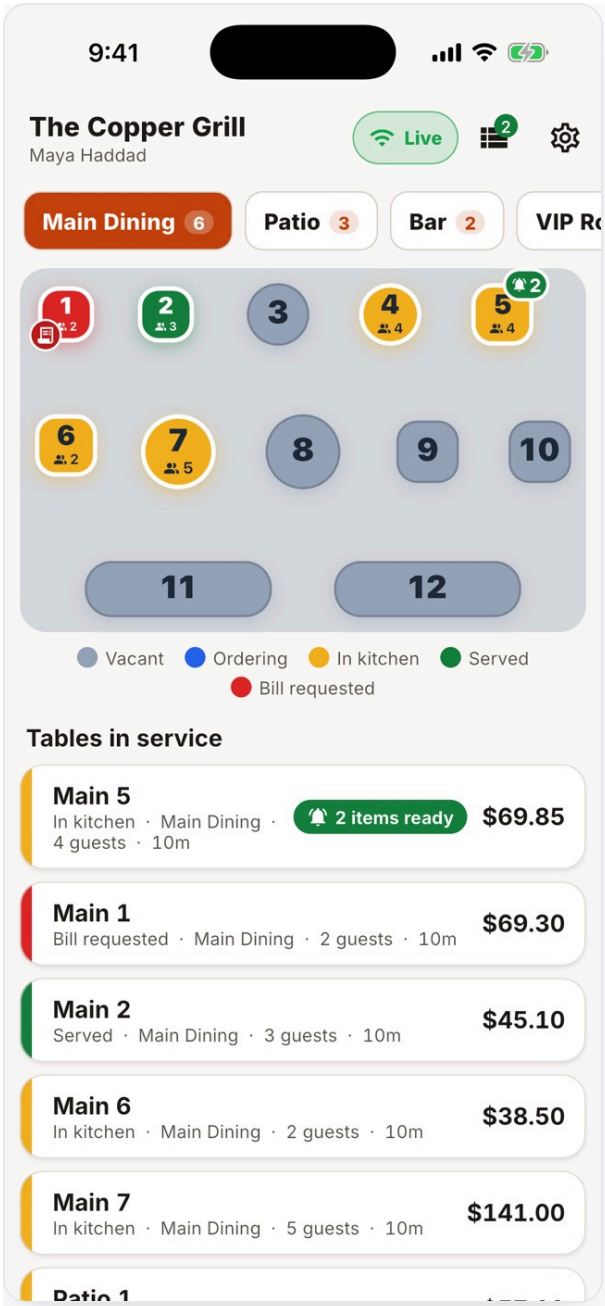


Kitchen, timers and the stock of the recipes.

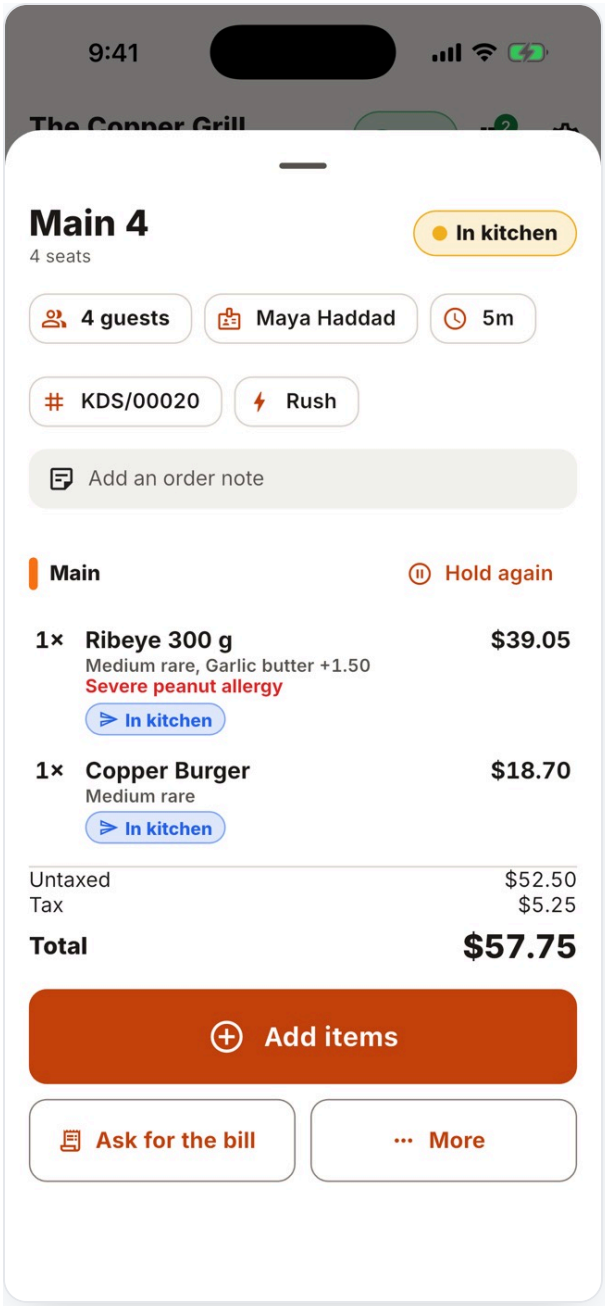
Setting	Meaning
Guests by default / Maximum	The guest stepper of the app starts at the default and stops at the maximum.
Hold and fire	Switch off if you never hold courses: everything is sent at once.
Merge tables / Split bills	Switch the features off for restaurants that do not need them.
Waiters collect payments	Off: only managers can take payments in the apps.
Waiters can mark sold out (86)	Off: only the kitchen and managers can.
Void after ready	Who can void an item that is already ready or served: only managers, waiters too, or nobody.
Stations / Default station	The screens of the restaurant and the one that gets dishes without a rule.
Warning / Late after (min)	The timer colours.
Consume ingredients, Recipe levels, Kitchen stock location	Chapter 11.
Product tips, tip choices	Chapter 12.
Receipt paper, print mode, copies, header, footer	58 or 80 mm; *Image* prints any language, the logo and the QR; *Text* is the fastest for Latin text.

15. The waiter app

In plain words: Sign in with the server address, your login and password. The app opens on the **floor map**: floors as tabs, every table in its live colour with the number of guests, the running total and the time since seating; a table pulses when food is ready.

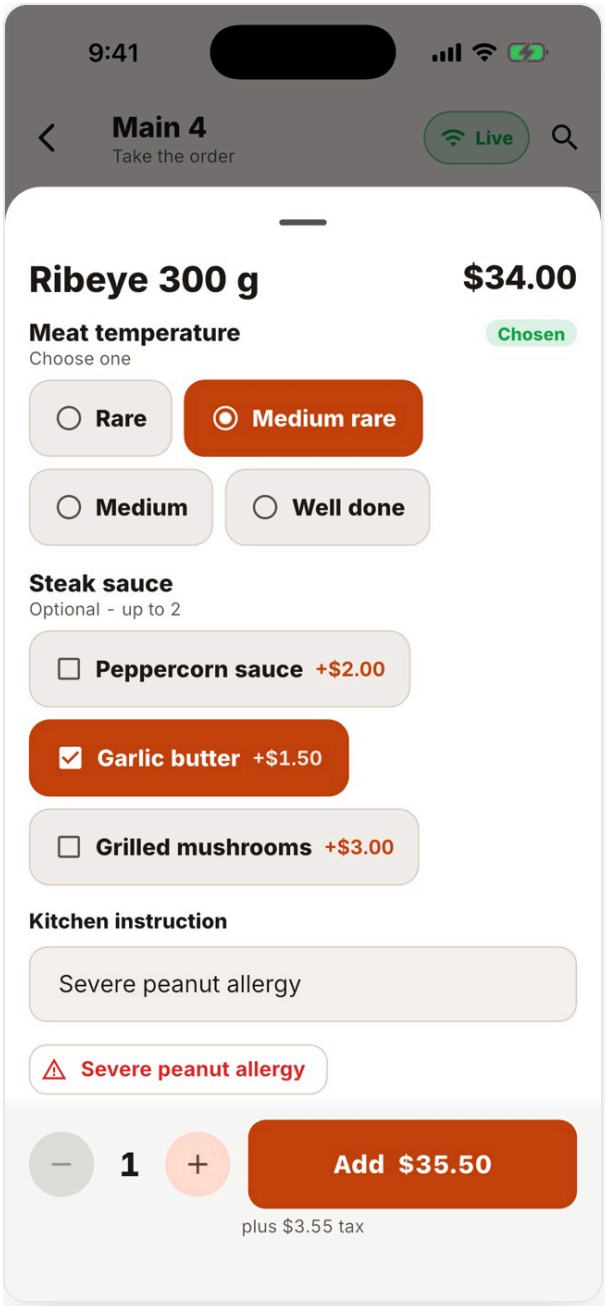


The floor map with live colours.

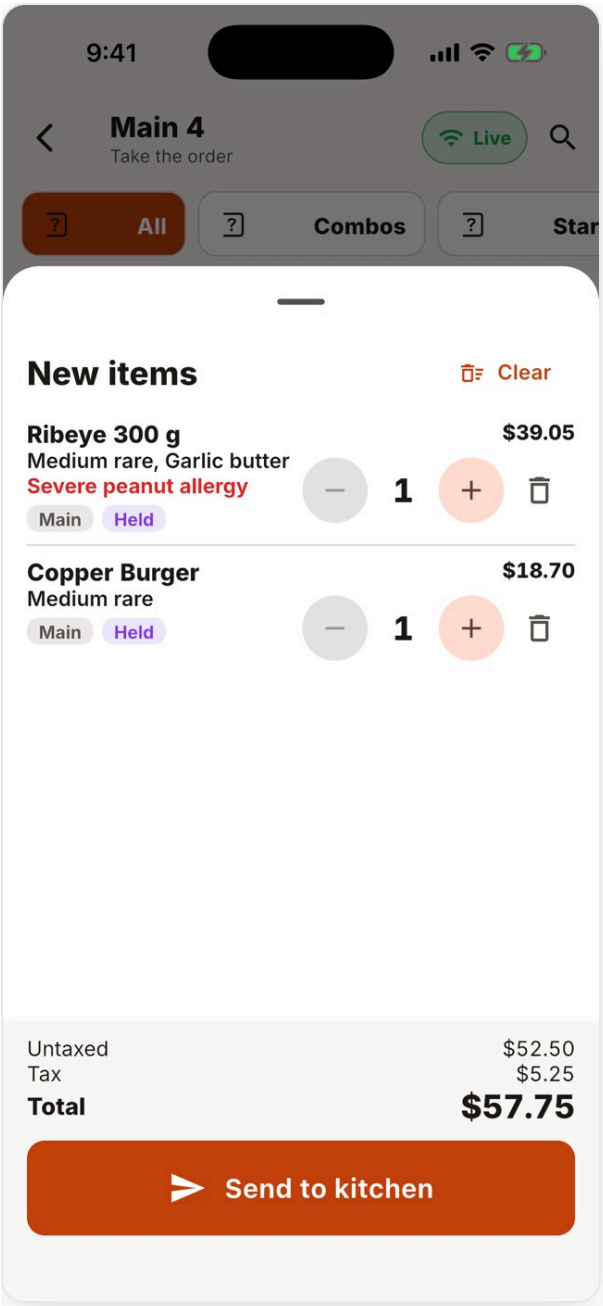


Tap a table: seat the guests, or see the order, its courses and the actions.

1. Tap a **grey** table, set the number of guests, press *Seat guests*.
2. Pick a category and tap a dish; use the search for a quick find. Sold-out dishes are greyed.
3. In the sheet that opens choose the **forced** options, the **add-ons**, the **seat** and the **course**, type a kitchen note, then *Add*.
4. Review the items in the cart, then press **Send to kitchen**. Held items stay on hold; everything else goes to the stations at once.



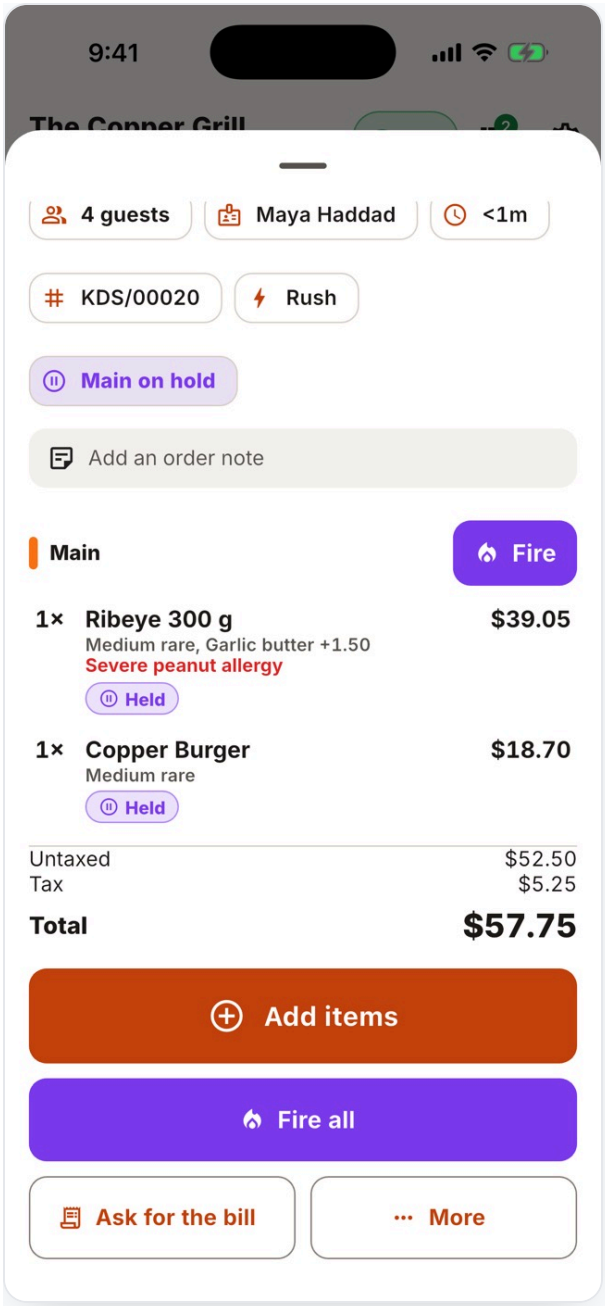
Forced options, paid add-ons, the kitchen note, the seat and the course in one sheet.



The cart with prices and the total.

16. Courses: hold and fire

In plain words: Dishes of a course on **hold** (mains and desserts by default) are saved but not sent. When the appetizers are cleared, press **Fire** on the course: the items go to their stations and the tickets appear. *Fire all* sends every held course, and *Hold* takes items that nobody started back.

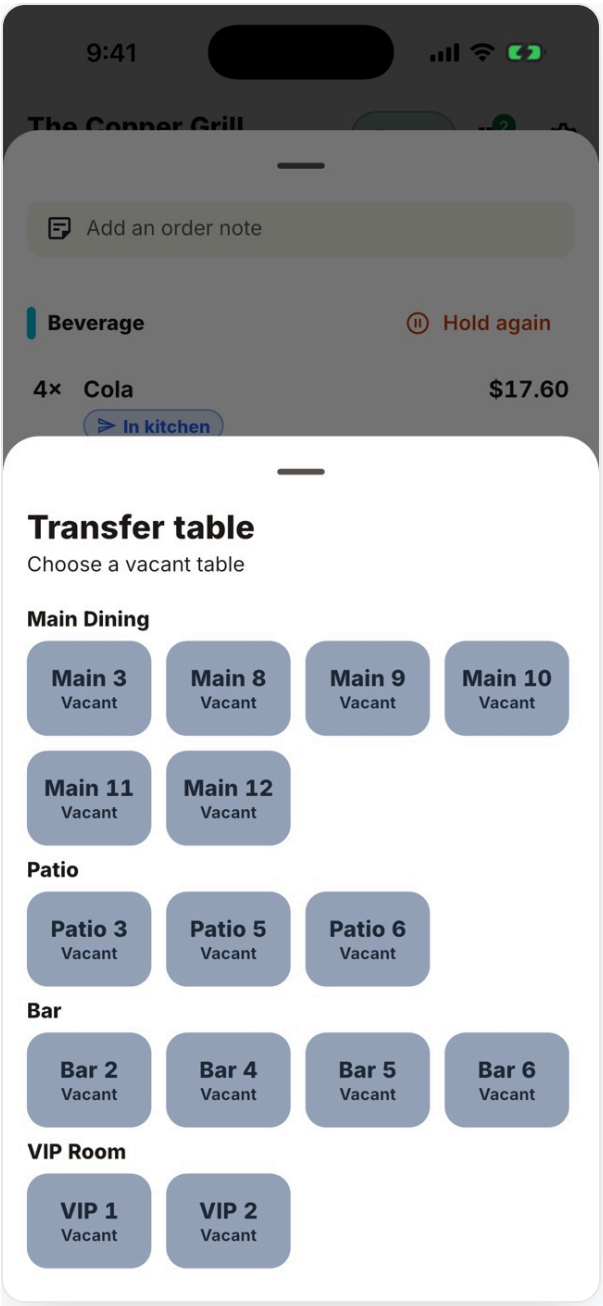


The order grouped by course with Fire buttons and the state of every item.

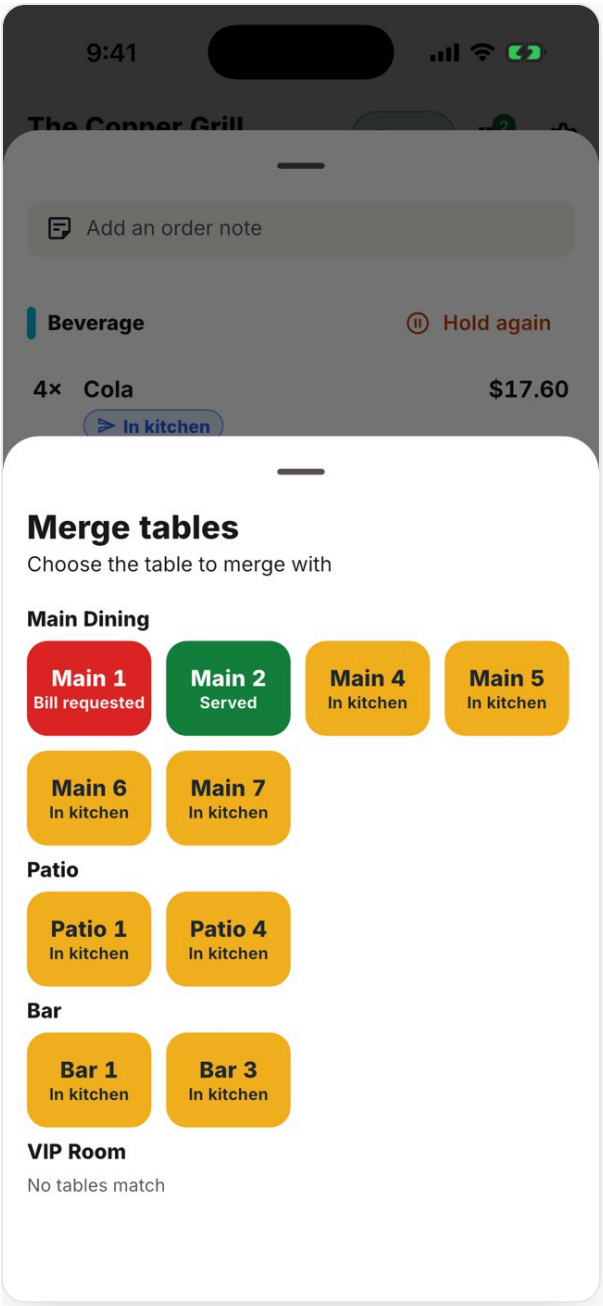
> **Tip:** Change the course of one dish before sending it, or switch *Hold* on or off for it in the add sheet. The bill cannot be asked while items are still on hold: fire or void them first.

17. Transfer, merge, void, rush and 86

Action	How	Result
Transfer	Table sheet → Transfer → pick a vacant table.	The party and the order move; the tickets show the new table.
Merge	Table sheet → Merge → pick the other table (occupied or not).	The tables share one order; both show the same colour. Unmerge when the party leaves.
Void	Long-press an item → Void → choose a reason.	The kitchen hears an alert and sees the item struck through. Items already ready can only be voided by managers (setting).
Rush	Order sheet → Rush.	Tickets of the order show a red RUSH tag and sort first.
Sold out (86)	Menu → long-press a dish → Sold out.	Every device greys the dish.



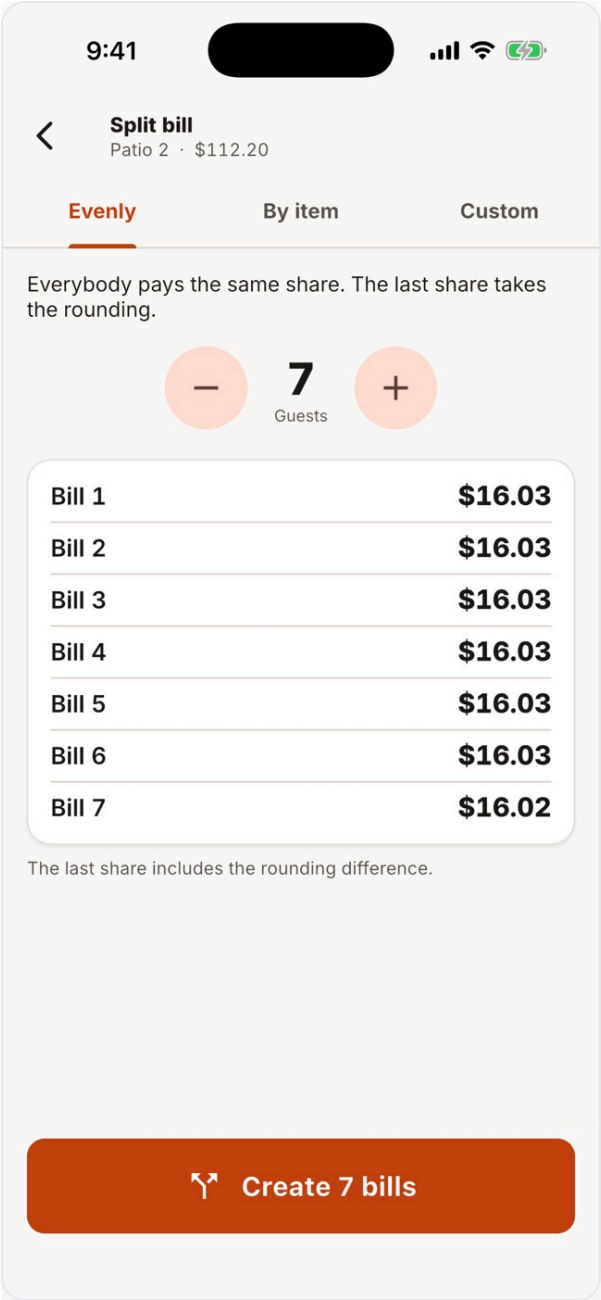
Choose the vacant table for the party.



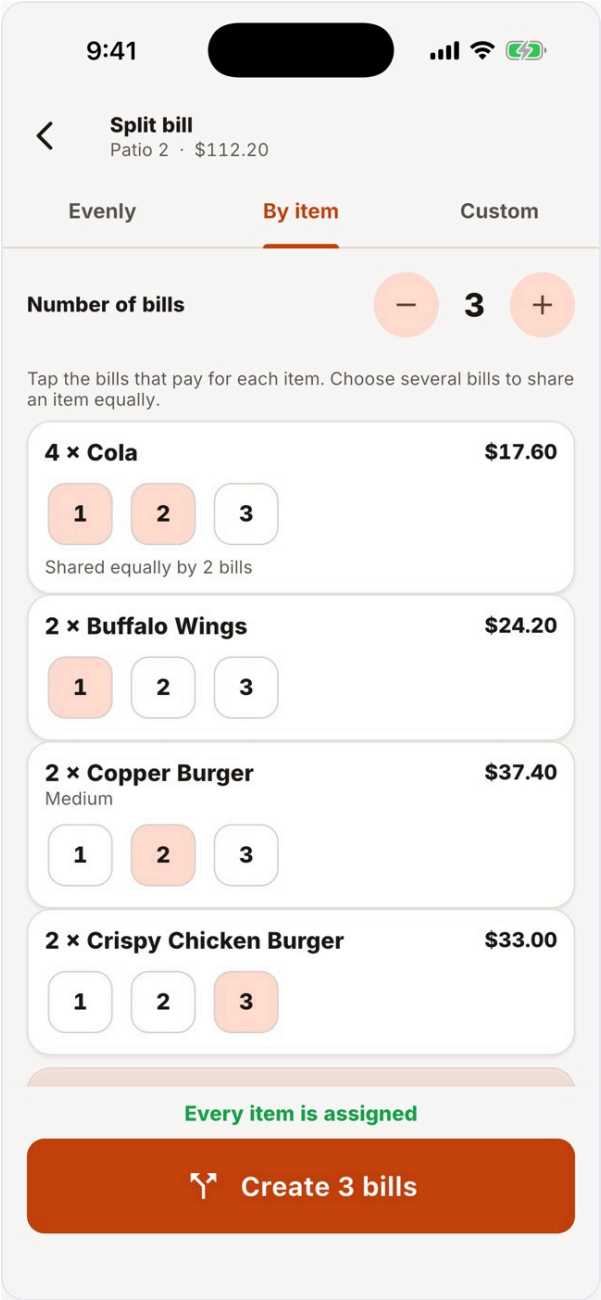
Merge two tables for a large party.

18. Splitting and paying the bill

In plain words: Ask for the bill: the table turns red and the cashiers are told. The **split sheet** offers three ways: *evenly* between guests, *by item or seat* (the app starts from the seat numbers of the items, you can give a share of a shared dish to several guests), and *custom amounts*. The sheet shows the total of each guest and warns if an item is not fully assigned.

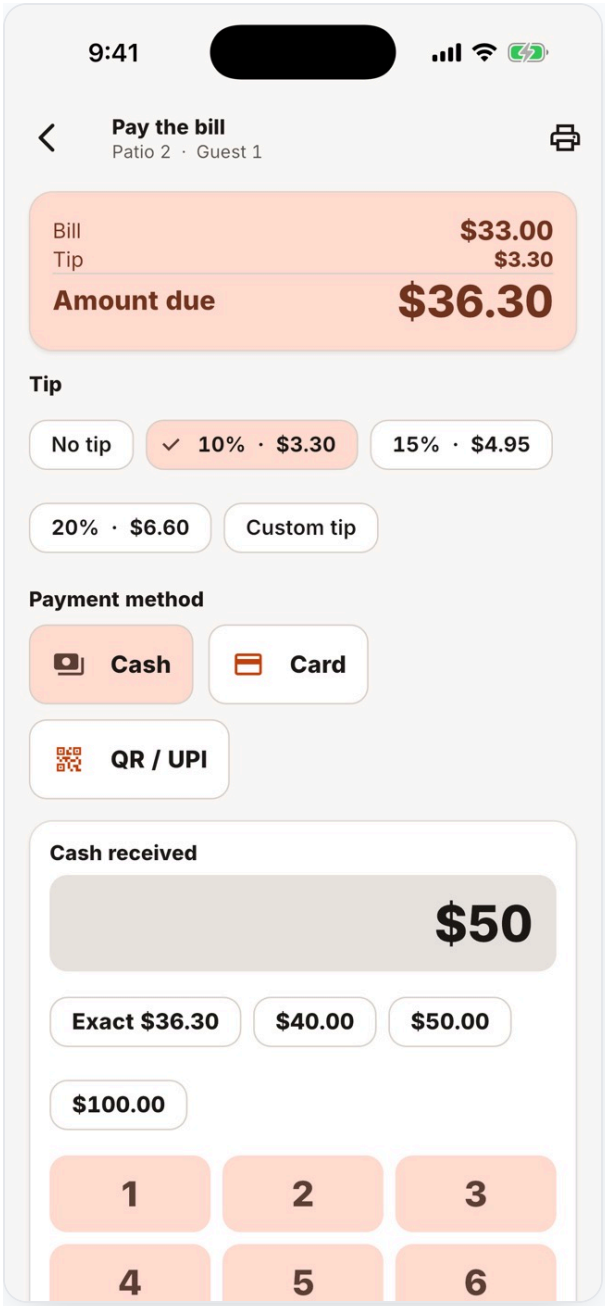


Even split: each guest's share; the last guest takes the rounding.

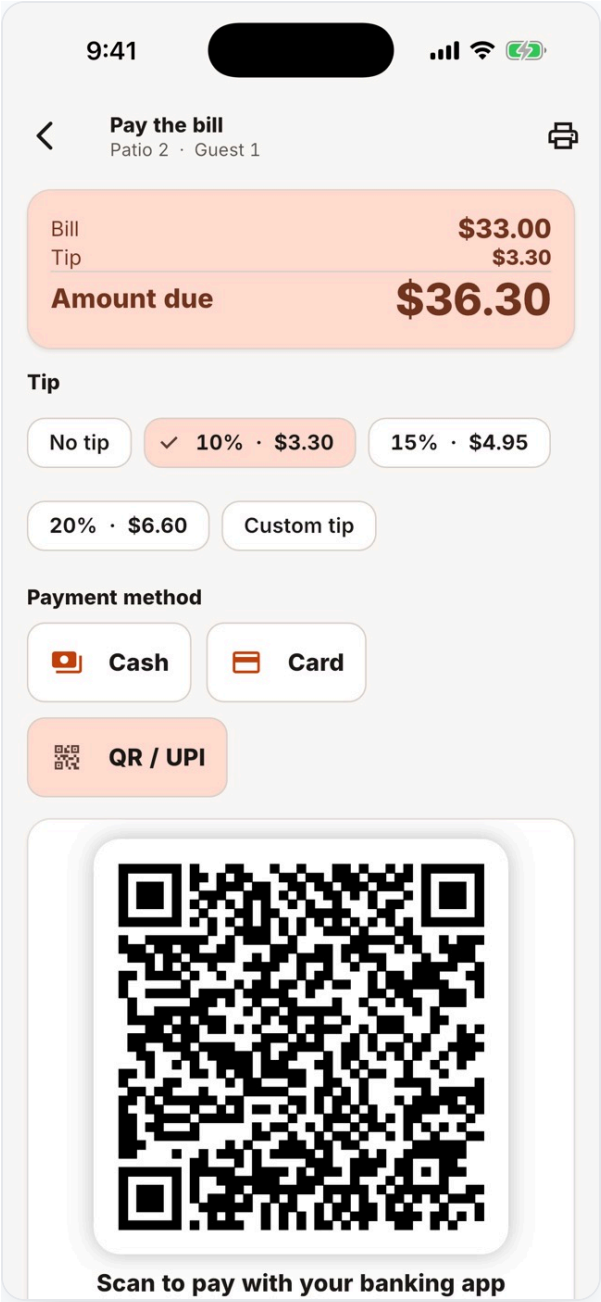


Split by item: each guest sees their own total.

In plain words: For each guest choose the payment: **cash** (the keypad and quick amounts show the change), **card** or other methods, or a **QR code** that shows the amount to scan. Add a tip, optionally the customer for an invoice, and press *Pay*. Then print the receipt or share it.



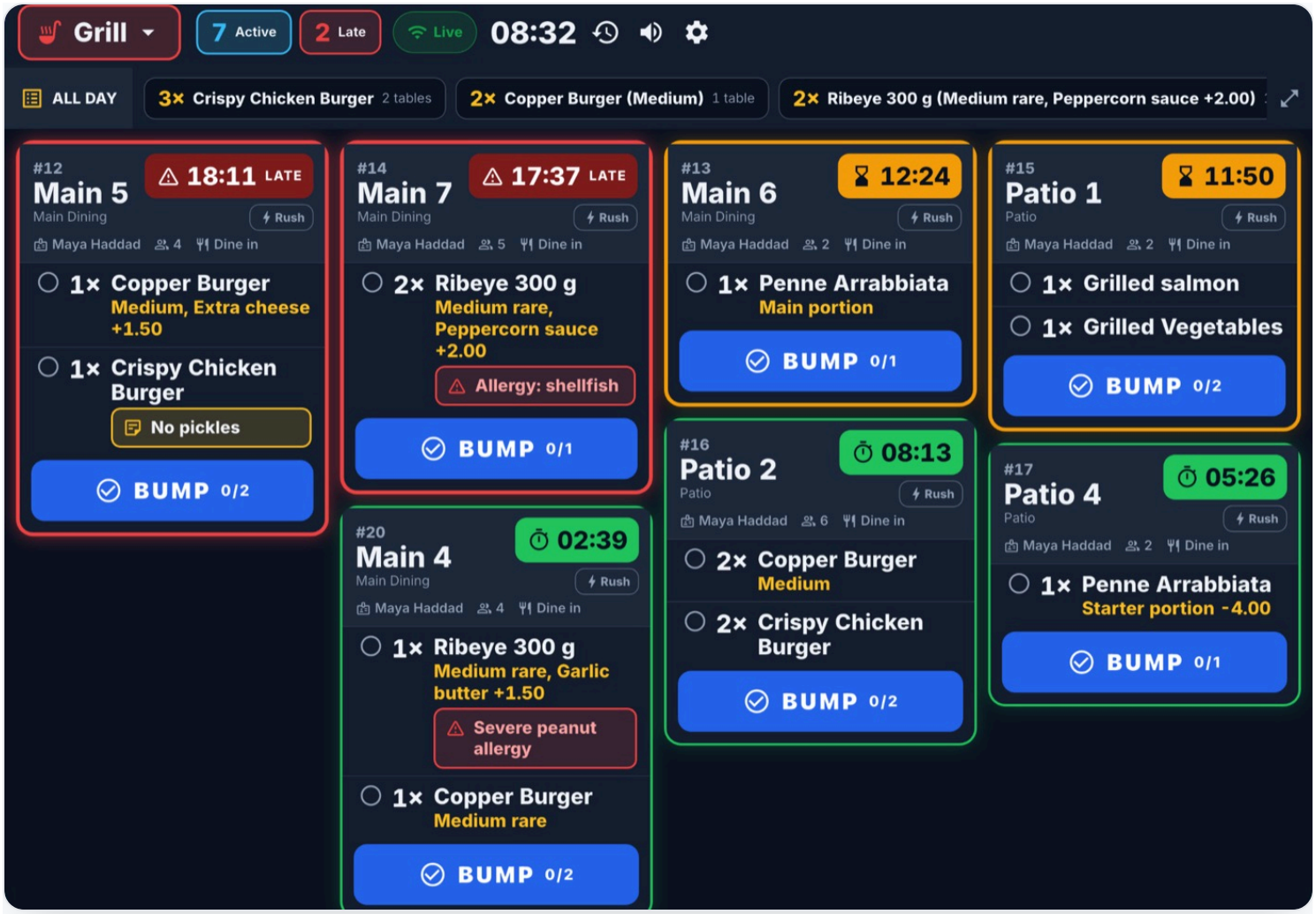
Cash with change due.



The QR of the bill for a digital payment.

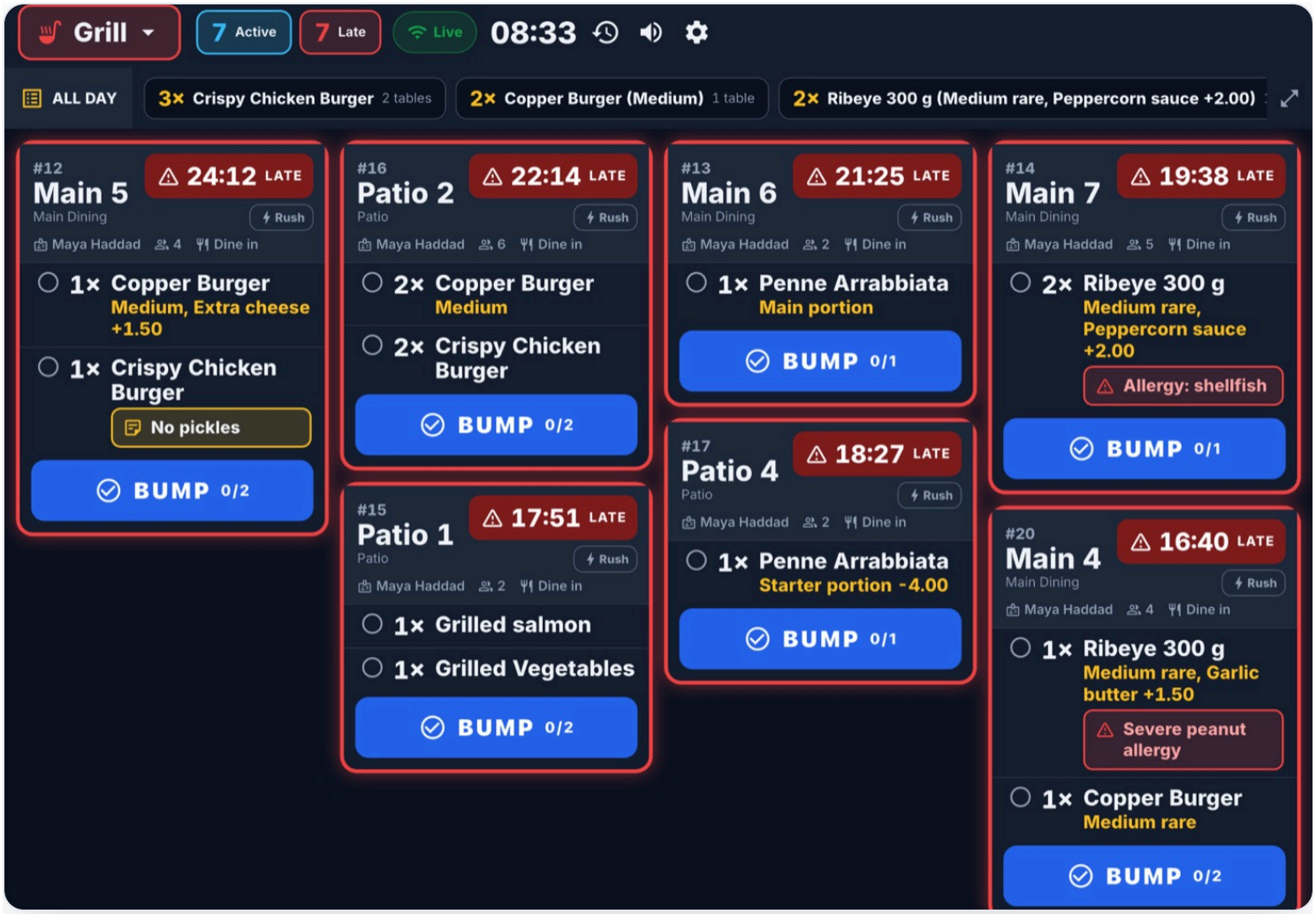
19. The kitchen display (KDS mode)

> **In plain words:** The KDS mode is made for a wall-mounted tablet: landscape, always awake, large text, dark theme. Choose the **station** at the top (Grill, Cold Prep, Bar or *Expo / All*); the choice is remembered on the device.

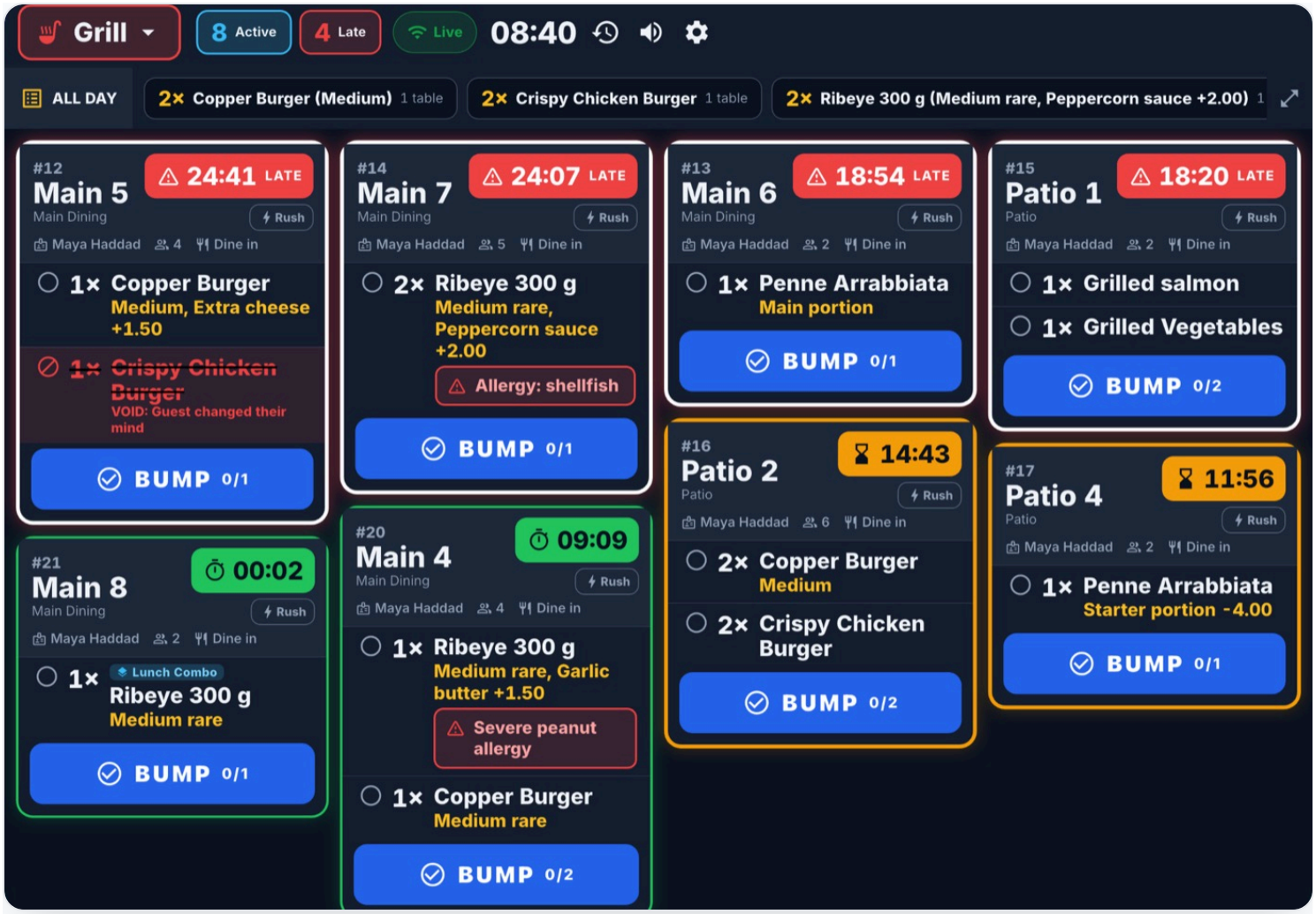


The board: oldest ticket first, items grouped by course, the SLA timer on every card.

On the card	What it shows
Header	Ticket number, table and floor, waiter, guests, a RUSH tag, an *add-on* tag when it is a second round.
Timer	Green, orange and flashing red (with an icon, not only colour) from the moment the ticket was sent.
Items	Quantity and name, the options in bold, the kitchen note in a highlighted box (allergies in red), the course.
Bump	One big button marks the whole ticket ready. A tap on an item marks only that item.



Late tickets turn orange, then flash red.

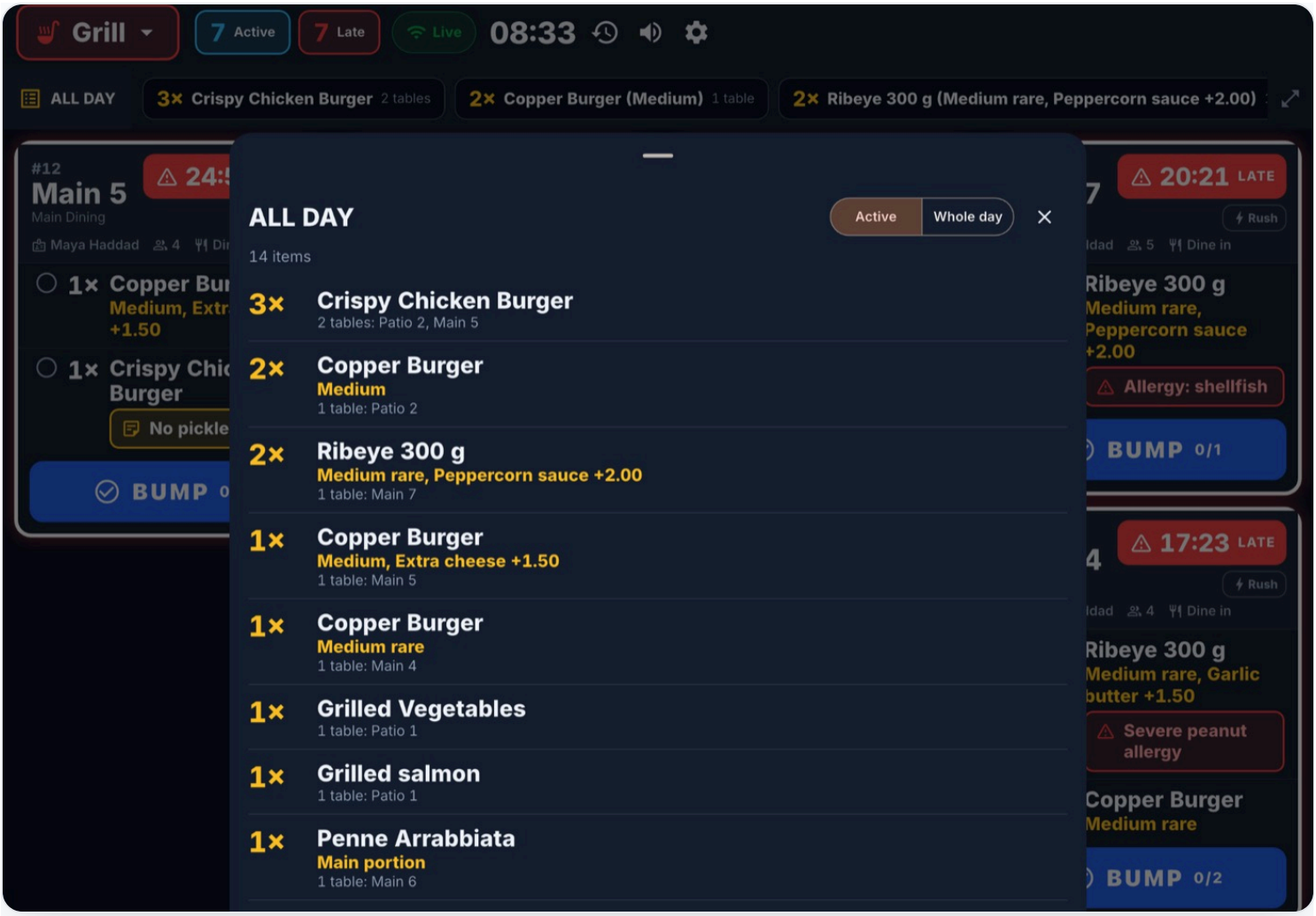


Items of a combo carry the combo name on the ticket.

> **In plain words: Sounds.** A chime plays for every new ticket and a different alert when an item is voided (the item is struck through in red). The sound can be muted in the settings.

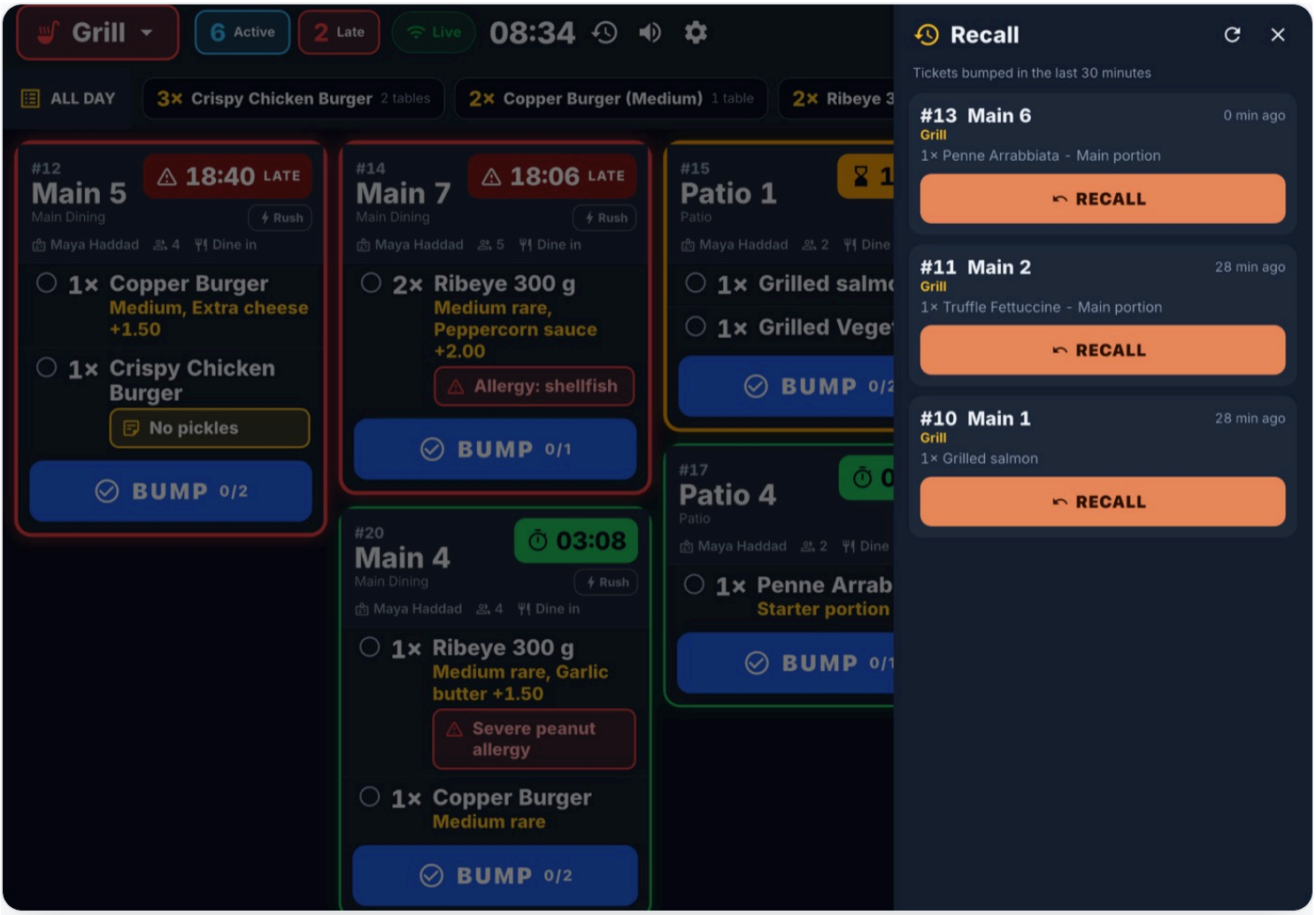
20. All-day count, recall and the expo view

> **In plain words:** The banner at the top is the **all-day / active item count**: identical items with identical options added up over all active tickets, for example *8 × Ribeye (Medium rare) across 4 tables*. A tap opens the full list, with a switch for the whole day.

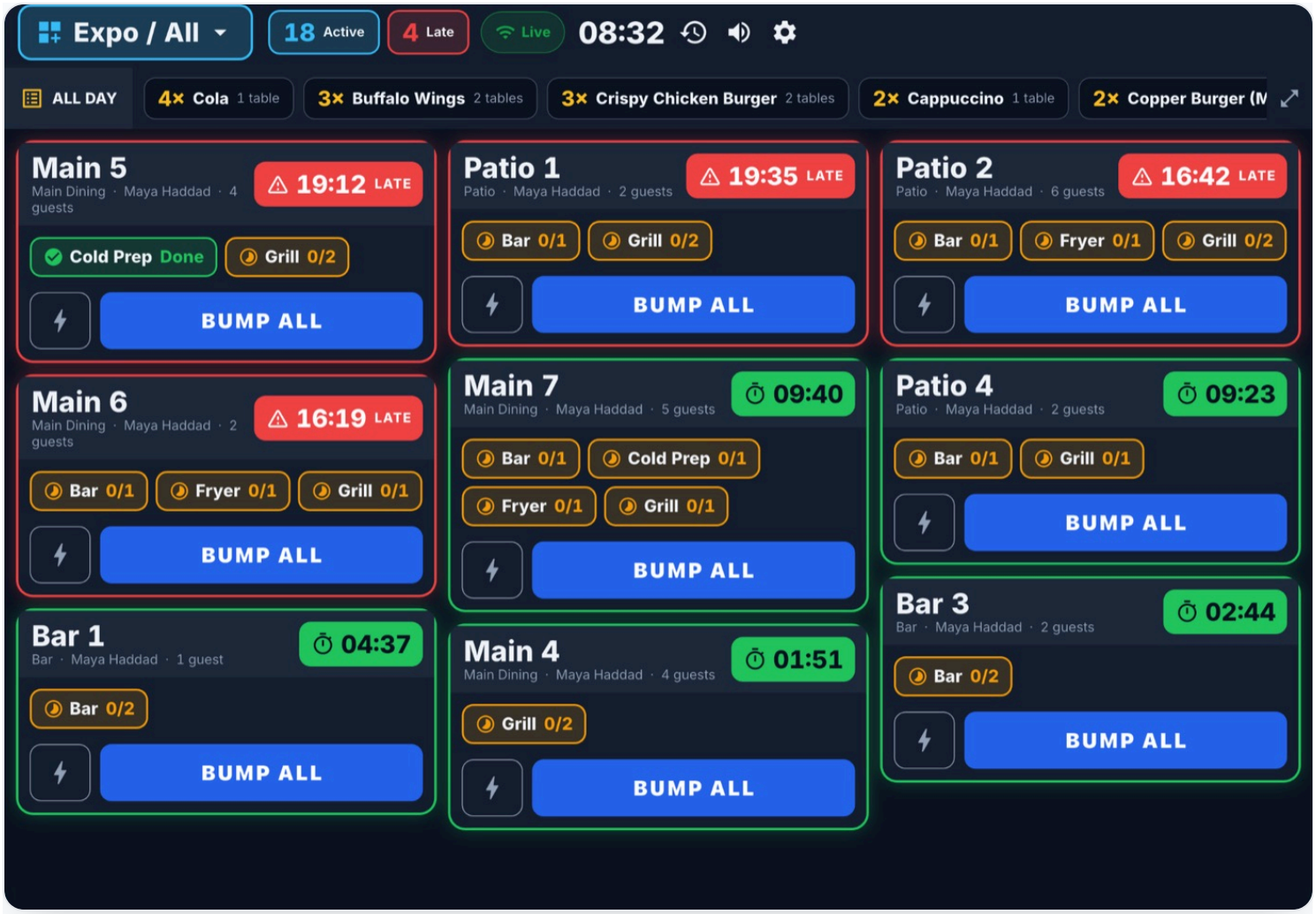


The item summary of the station.

> **In plain words: Recall.** A ticket bumped by mistake is in the *Undo / Recall* drawer for 30 minutes; one tap brings it back with its original time. **Expo.** The expo view shows every station's tickets grouped by table, with a chip per station, so the expediter sees when the whole table is complete.



The recall drawer.



The expo view: one card per table, one chip per station.

21. Real time on your own network

> **In plain words:** Every change (a new ticket, a bump, a void, a table colour) is written to an **event log** in Odoo and pushed on the **Odoo websocket** on a secret channel of the restaurant. All devices subscribe to it. Measured on a development laptop, a ticket appears on the websocket 2 to 4 ms after Odoo answered the waiter, about 80 ms from the tap to the screen including the HTTP request; on a real LAN count on well under 300 ms. The numbers depend on your Wi-Fi and server.

Situation	What happens
The websocket drops	The kitchen board keeps the last tickets, timers keep running, a banner says *reconnecting*. When it is back the app asks for the events it missed (`/events?since=`); if the log no longer has them it reloads everything.
The server cannot be reached by a waiter	The orders are kept in the outbox of the phone (with an unchangeable sequence number and a unique ID) and sent in order when the link returns; the table shows *not sent yet*.
The internet is down	Nothing changes if Odoo runs in the restaurant: the devices only need the local network. If Odoo is in the cloud, the service stops, so use a local server for the live service.
Websocket blocked by a proxy	The app falls back to polling the event log every 15 seconds.
Two waiters send to one table	Both orders are added to the same open order.

In plain words: Installing the server on the LAN. Run Odoo and PostgreSQL on a small computer in the restaurant, give it a fixed IP address (for example `192.168.1.10`), use the Odoo address `http://192.168.1.10:8069` in the app, and keep the cloud for nightly backups. With `workers = 0` (default for small installs) the websocket is on the same port. With workers, run the gevent port (`gevent_port = 8072`) and route `/websocket` to it:

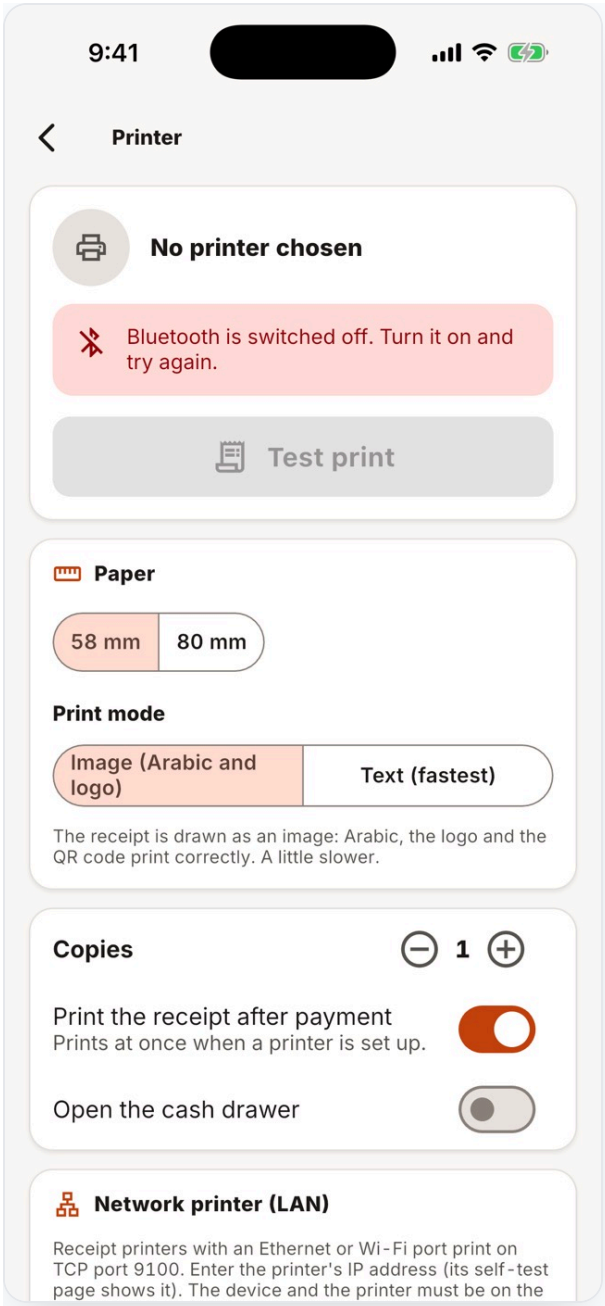
```
location /websocket {
    proxy_pass http://127.0.0.1:8072;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_read_timeout 3600s;
}
```

Important: Plain `http` on a local network is allowed by the app builds only when you enable it (the setup script has an option; the trade-off is explained in the app README). Use `https` whenever the server is on the internet.

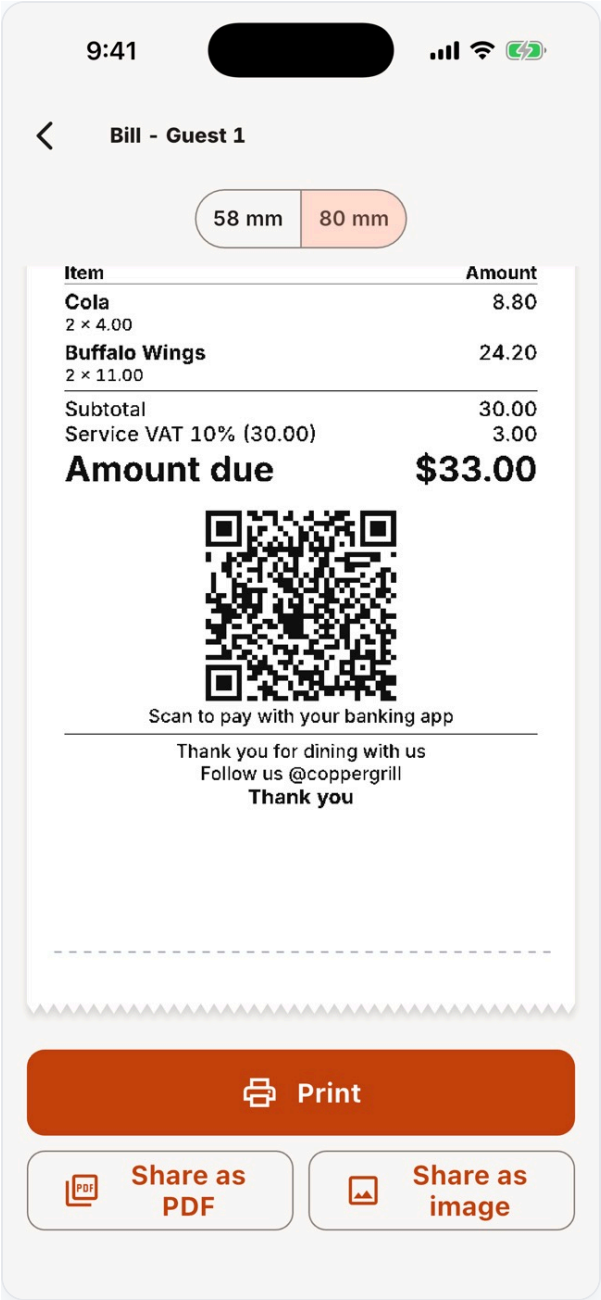
---

22. Printing receipts

In plain words: Receipts are printed by the **app**: no printing hardware needs to be attached to Odoo. The app generates the ESC/POS commands itself and sends them over **Bluetooth** (Android: Bluetooth Classic printers; iPhone: only printers with Bluetooth Low Energy) or over the **network** (TCP, usually port 9100). Pair the printer in *Settings* → *Printer*; the test print checks the width (58 or 80 mm).



Choose the printer and the paper width.



The receipt as printed on 80 mm: items with options, taxes, payments, tip and QR.

In plain words: Print mode. *Image* (default) draws the receipt as a picture at the printer's width, so Arabic, the logo and the QR code print correctly; *Text* sends plain ESC/POS text, the fastest for Latin text. A pro-forma bill can be printed before payment, and the receipt can be shared as PDF or image. The same receipt exists in Odoo as an 80 mm report on every bill.

The Copper Grill

21 Harbour Street
+971 4 555 0142
VAT: 100234567800003
Fire-grilled steaks since 2015
Open daily 12:00 to midnight

Table: **VIP 2** · Guests: 3
Order: KDS/00060 (#1)
Waiter: Noah Rahman
Date: 10/27/2026 04:54:17 AM
Bill: Full bill

1.00 x Caesar Salad	14.30
+ On the salad	
1.00 x Penne Arrabbiata	16.50
+ Main portion	
1.00 x Burrata & Tomato	15.40
1.00 x Truffle Fettuccine	20.90
+ Main portion	
1.00 x Caesar Salad	14.30
+ On the salad	
1.00 x Penne Arrabbiata	16.50
+ Main portion	
1.00 x House Red (glass)	10.35
1.00 x Tiramisu	8.80
2.00 x Truffle Fries	13.20

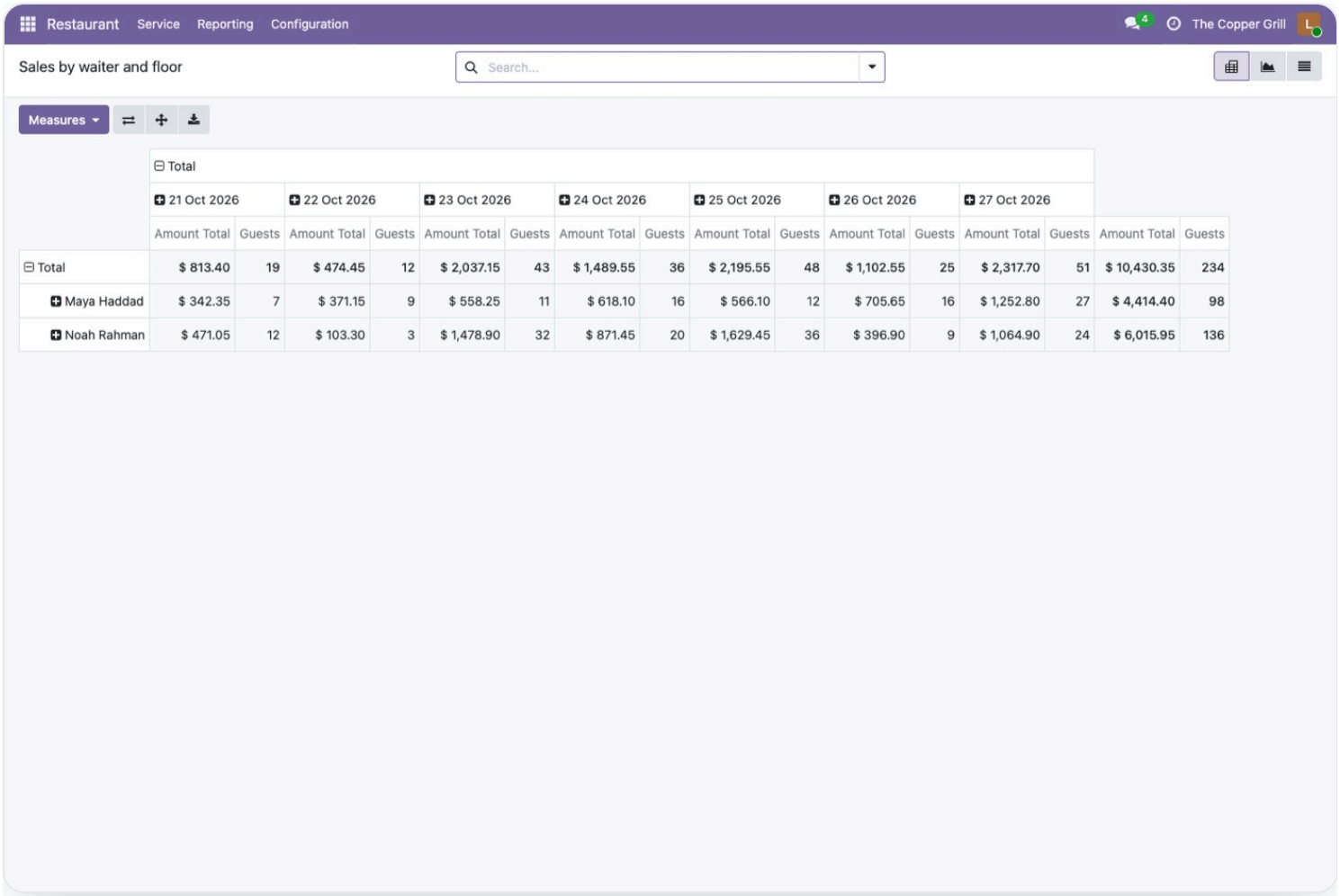
Untaxed	$ 118.00
Service VAT 10%	$ 10.90
15%	$ 1.35
Tip	$ 19.54
TOTAL	**$ 149.79**
Card	$ 149.79

Thank you for dining with us
Follow us @coppergrill

The 80 mm receipt report in Odoo.

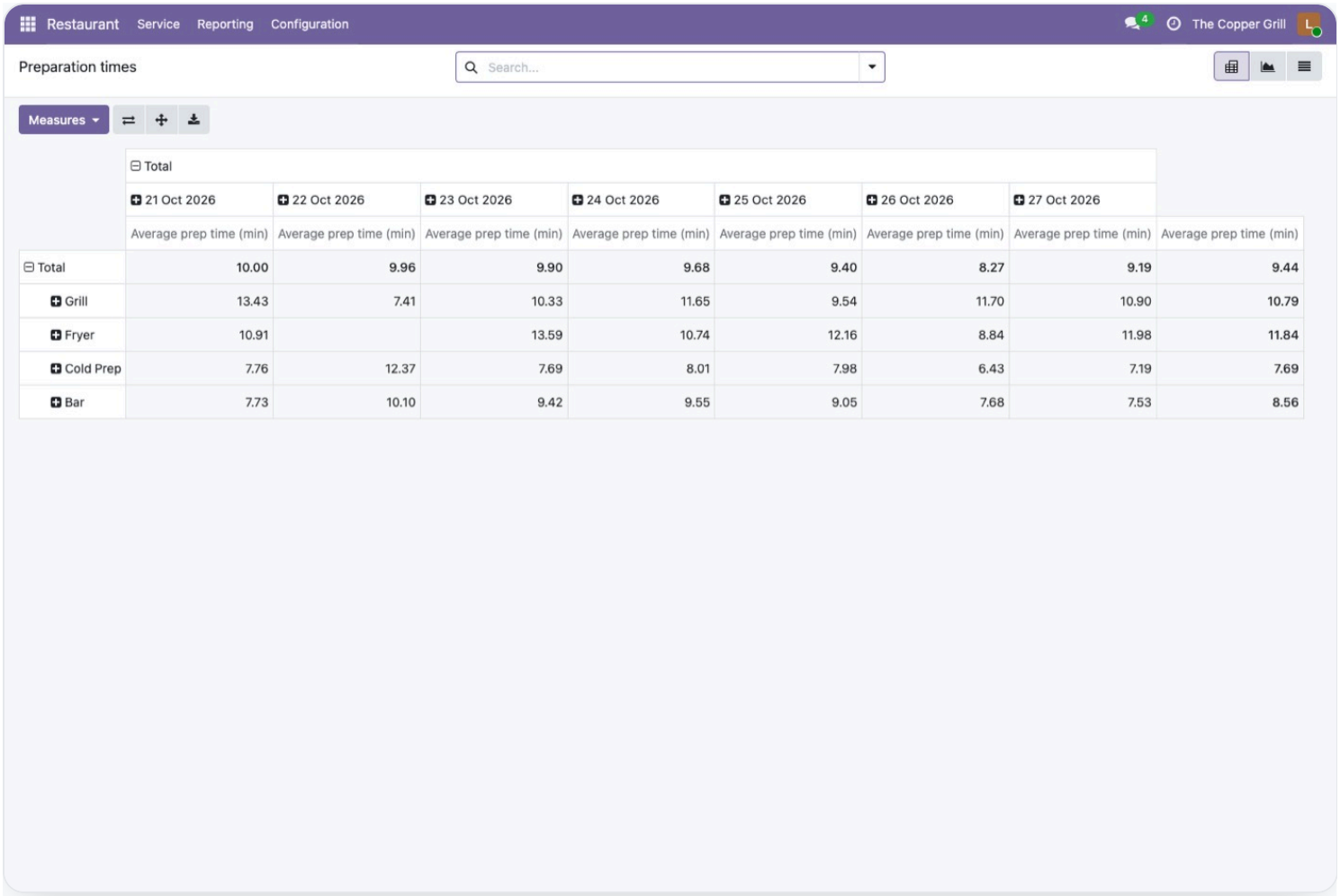
23. Reports

> **In plain words:** Restaurant → Reporting has three analyses, each with pivot, graph and list views and the usual filters and groupings:



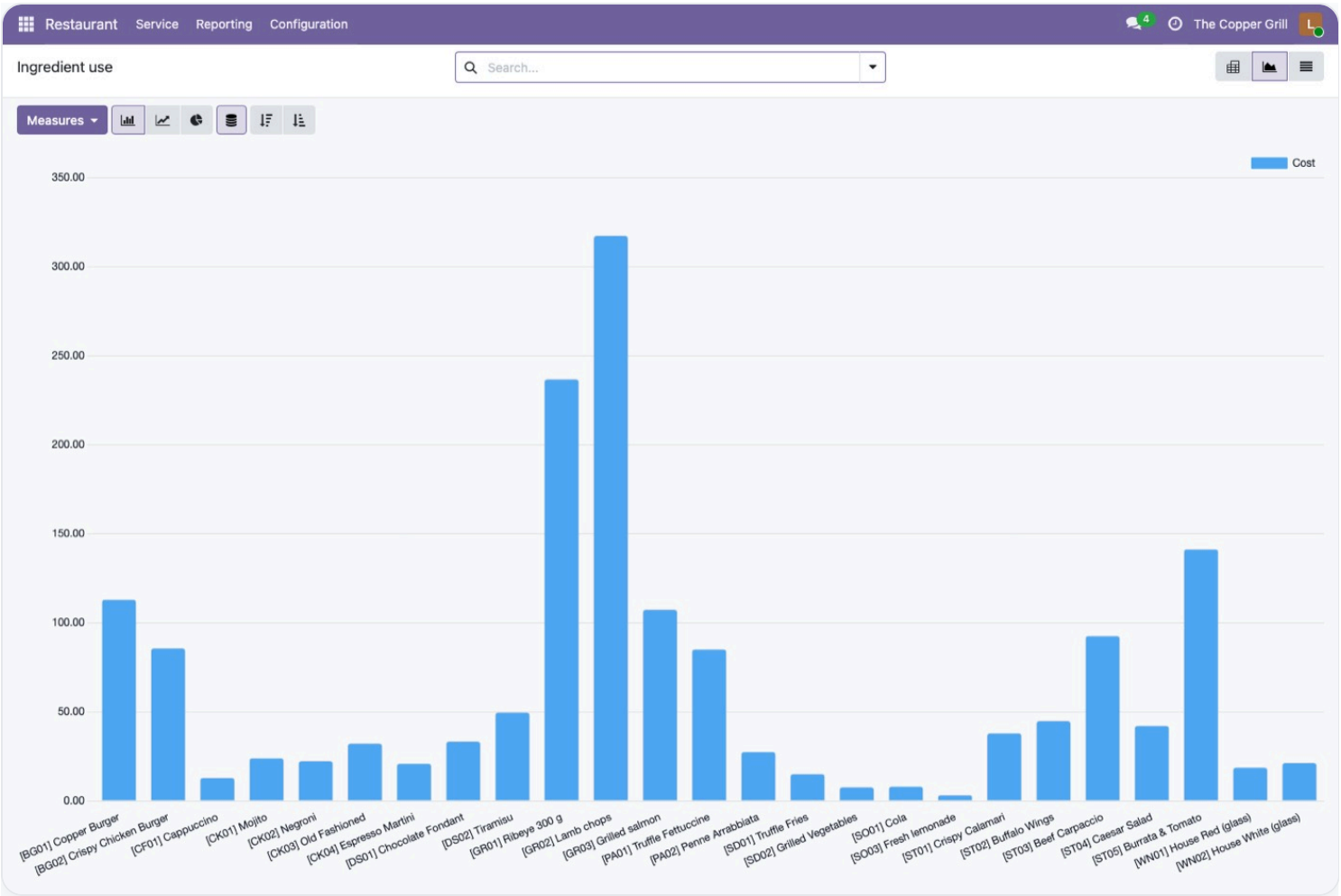
	Total															
	21 Oct 2026		22 Oct 2026		23 Oct 2026		24 Oct 2026		25 Oct 2026		26 Oct 2026		27 Oct 2026			
	Amount Total	Guests	Amount Total	Guests	Amount Total	Guests	Amount Total	Guests	Amount Total	Guests	Amount Total	Guests	Amount Total	Guests	Amount Total	Guests
Total	$ 813.40	19	$ 474.45	12	$ 2,037.15	43	$ 1,489.55	36	$ 2,195.55	48	$ 1,102.55	25	$ 2,317.70	51	$ 10,430.35	234
Maya Haddad	$ 342.35	7	$ 371.15	9	$ 558.25	11	$ 618.10	16	$ 566.10	12	$ 705.65	16	$ 1,252.80	27	$ 4,414.40	98
Noah Rahman	$ 471.05	12	$ 103.30	3	$ 1,478.90	32	$ 871.45	20	$ 1,629.45	36	$ 396.90	9	$ 1,064.90	24	$ 6,015.95	136

Sales by waiter and by day (paid orders).



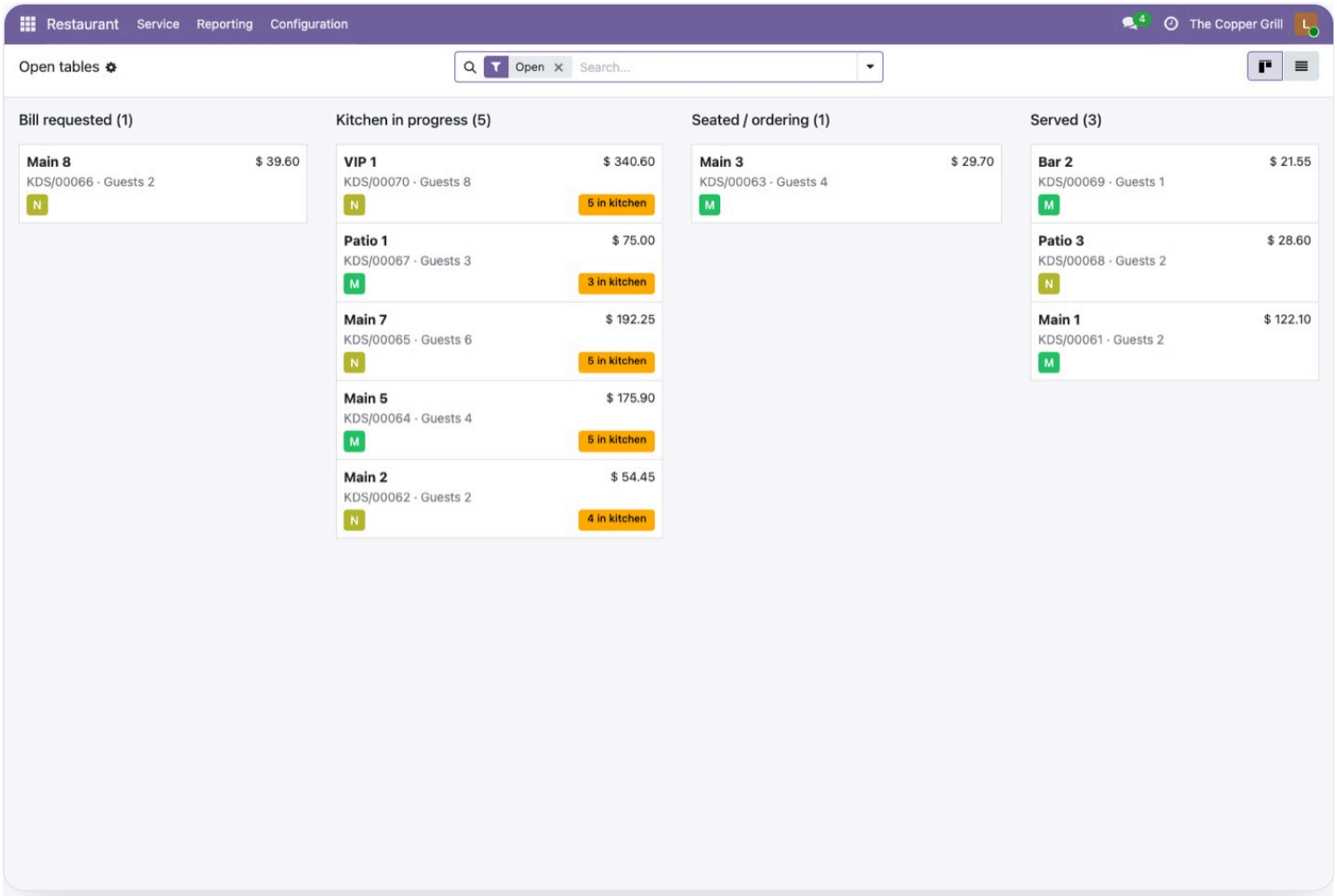
	Total							
	21 Oct 2026	22 Oct 2026	23 Oct 2026	24 Oct 2026	25 Oct 2026	26 Oct 2026	27 Oct 2026	
	Average prep time (min)	Average prep time (min)	Average prep time (min)	Average prep time (min)	Average prep time (min)	Average prep time (min)	Average prep time (min)	Average prep time (min)
Total	10.00	9.96	9.90	9.68	9.40	8.27	9.19	9.44
Grill	13.43	7.41	10.33	11.65	9.54	11.70	10.90	10.79
Fryer	10.91		13.59	10.74	12.16	8.84	11.98	11.84
Cold Prep	7.76	12.37	7.69	8.01	7.98	6.43	7.19	7.69
Bar	7.73	10.10	9.42	9.55	9.05	7.68	7.53	8.56

Preparation times by station and day; filter on late tickets.

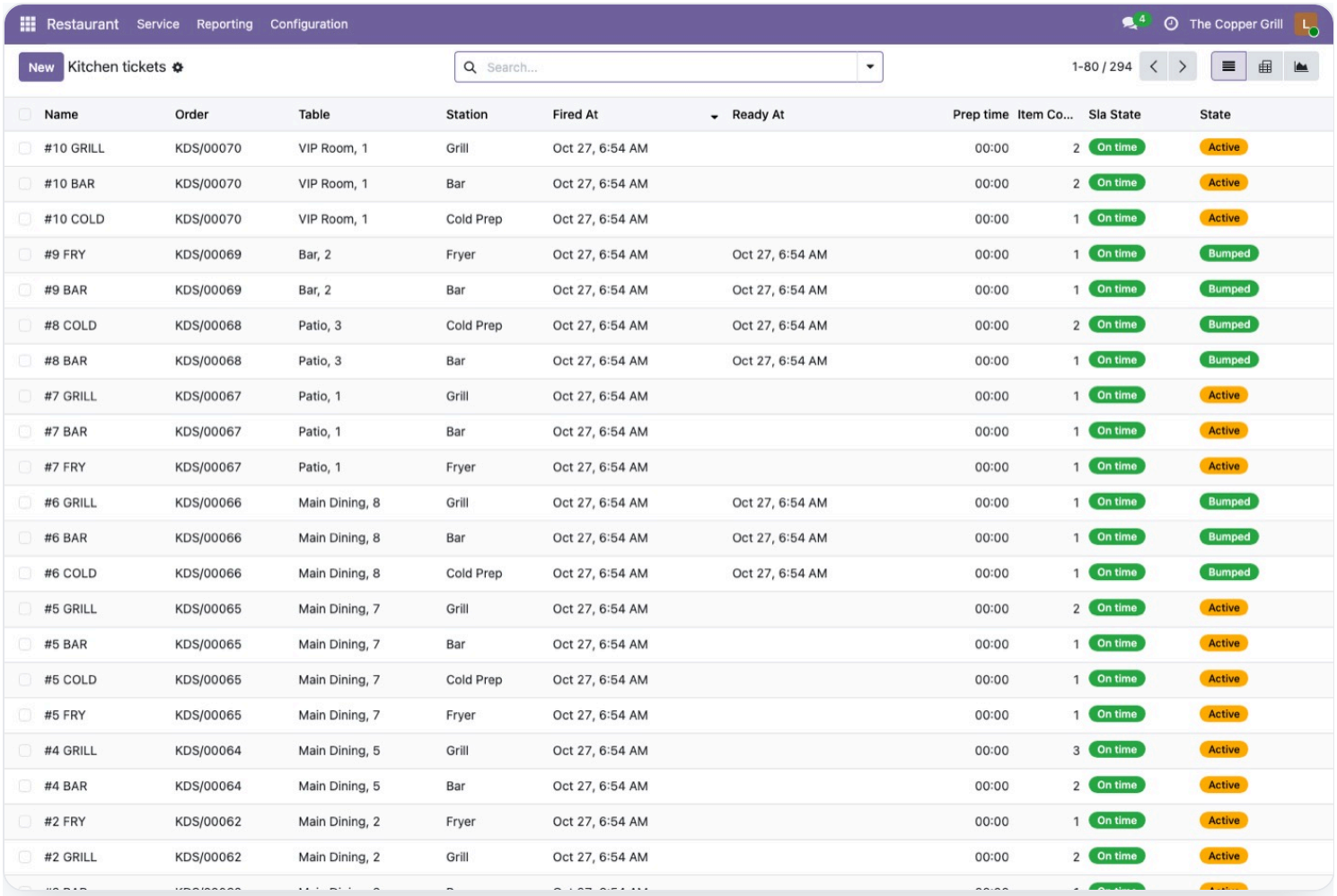


Food cost by dish from the ingredient use.

> **In plain words:** **Open tables** is a kanban of the live dining room by table state, **Kitchen tickets** lists every ticket with its preparation time and SLA state, and **Bills** lists every bill with its payments, tip and invoice.



Open tables by state.



Name	Order	Table	Station	Fired At	Ready At	Prep time	Item Co...	Sla State	State
#10 GRILL	KDS/00070	VIP Room, 1	Grill	Oct 27, 6:54 AM		00:00	2	On time	Active
#10 BAR	KDS/00070	VIP Room, 1	Bar	Oct 27, 6:54 AM		00:00	2	On time	Active
#10 COLD	KDS/00070	VIP Room, 1	Cold Prep	Oct 27, 6:54 AM		00:00	1	On time	Active
#9 FRY	KDS/00069	Bar, 2	Fryer	Oct 27, 6:54 AM	Oct 27, 6:54 AM	00:00	1	On time	Bumped
#9 BAR	KDS/00069	Bar, 2	Bar	Oct 27, 6:54 AM	Oct 27, 6:54 AM	00:00	1	On time	Bumped
#8 COLD	KDS/00068	Patio, 3	Cold Prep	Oct 27, 6:54 AM	Oct 27, 6:54 AM	00:00	2	On time	Bumped
#8 BAR	KDS/00068	Patio, 3	Bar	Oct 27, 6:54 AM	Oct 27, 6:54 AM	00:00	1	On time	Bumped
#7 GRILL	KDS/00067	Patio, 1	Grill	Oct 27, 6:54 AM		00:00	1	On time	Active
#7 BAR	KDS/00067	Patio, 1	Bar	Oct 27, 6:54 AM		00:00	1	On time	Active
#7 FRY	KDS/00067	Patio, 1	Fryer	Oct 27, 6:54 AM		00:00	1	On time	Active
#6 GRILL	KDS/00066	Main Dining, 8	Grill	Oct 27, 6:54 AM	Oct 27, 6:54 AM	00:00	1	On time	Bumped
#6 BAR	KDS/00066	Main Dining, 8	Bar	Oct 27, 6:54 AM	Oct 27, 6:54 AM	00:00	1	On time	Bumped
#6 COLD	KDS/00066	Main Dining, 8	Cold Prep	Oct 27, 6:54 AM	Oct 27, 6:54 AM	00:00	1	On time	Bumped
#5 GRILL	KDS/00065	Main Dining, 7	Grill	Oct 27, 6:54 AM		00:00	2	On time	Active
#5 BAR	KDS/00065	Main Dining, 7	Bar	Oct 27, 6:54 AM		00:00	1	On time	Active
#5 COLD	KDS/00065	Main Dining, 7	Cold Prep	Oct 27, 6:54 AM		00:00	1	On time	Active
#5 FRY	KDS/00065	Main Dining, 7	Fryer	Oct 27, 6:54 AM		00:00	1	On time	Active
#4 GRILL	KDS/00064	Main Dining, 5	Grill	Oct 27, 6:54 AM		00:00	3	On time	Active
#4 BAR	KDS/00064	Main Dining, 5	Bar	Oct 27, 6:54 AM		00:00	2	On time	Active
#2 FRY	KDS/00062	Main Dining, 2	Fryer	Oct 27, 6:54 AM		00:00	1	On time	Active
#2 GRILL	KDS/00062	Main Dining, 2	Grill	Oct 27, 6:54 AM		00:00	2	On time	Active

Every ticket with its SLA state.

24. Build and publish the apps

> **In plain words:** Unzip `flutter-app-source.zip`. The README in the folder is the full guide; the short version:

1. Install Flutter 3.41 or newer (`flutter doctor` must be green for your platforms).
2. Run the setup script with your server and your package name, and optionally the mode to lock the device to:

```
dart run tools/setup.dart --url http://192.168.1.10:8069 --package com.yourrestaura
```

3. Replace the placeholder icon (`assets/branding`), run `flutter pub get`.

4. Android: `flutter build apk --release` (or `appbundle` for Google Play). iPhone and iPad: `flutter build ios --release`, then archive in Xcode.
5. Install the waiter build on the phones and the same app on the kitchen tablets: the mode follows the user's role, or lock a tablet to *KDS* with the setup script.
6. Kitchen tablets: use the device's guided access / kiosk mode so the app cannot be closed, and keep them powered.

> **Important:** The app can be built without any store account for your own devices (a debug or ad-hoc build). The stores are only needed to distribute to many devices easily.

---

25. API for developers

> **In plain words:** Everything the apps do is available over a small REST API (`/api/kds/v1`) plus the live websocket. The contract is in `documentation/API.md`. Operations are idempotent; a bump or a payment sent twice has the effect of one.

```
# is the server reachable?
curl http://192.168.1.10:8069/api/kds/v1/ping
# {"ok": true, "api_version": 1, "server_time": "2026-10-27T18:30:00Z"}

# sign in (returns a token and the restaurants of the account)
curl -X POST http://192.168.1.10:8069/api/kds/v1/auth/login -H 'Content-Type: application
  -d '{"login": "maya@coppergrill.example", "password": "...", "device": {"id": "tablet-1

# send an order (idempotent: the same operation uuid is applied once)
curl -X POST http://192.168.1.10:8069/api/kds/v1/op/order_push -H "Authorization: Bearer
  -d '{"uuid": "4c1f...", "order_uuid": "a9d2...", "table_id": 12, "guests": 4, "fire": t
       "lines": [{"uuid": "7b0e...", "product_id": 82, "qty": 2, "option_ids": [31], "not

# watch the kitchen events live (Odoo websocket; send an Origin header)
#   ws://192.168.1.10:8069/websocket   then   {"event_name":"subscribe","data":{"channels
```

Call	Use
`POST /auth/login`	Token, user roles and restaurants.
`GET /bootstrap`	Menu, modifiers, courses, stations, floors, open orders, real-time settings.
`POST /op/<type>`, `POST /sync`	One operation or an ordered batch of up to 100 (order_push, course_fire, ticket_bump, bill_pay, ...).
`GET /tickets`, `/summary`, `/tickets/history`	The board of a station, the all-day count and the recall drawer.
`GET /events?since=`	Catch up after a dropout.
`GET /bills/<id>/receipt`	Everything needed to print a receipt.

26. Update, backup and uninstall

In plain words: Replace the module folder with the new version, restart Odoo and run `-u taqnix_restaurant_kds`. Your orders, bills, tickets and settings are kept. Update the apps with the new source and rebuild; the API is versioned (`/v1`), and older apps keep working with newer modules within the same version.

In plain words: Back up the database and the filestore as usual. The live event log keeps only 48 hours and is cleaned automatically. To uninstall, use Apps → Restaurant Waiter App → Uninstall; the POS orders and invoices created for paid bills stay in Odoo.

27. Troubleshooting

Problem	Cause and fix
The app says *no restaurant is set up*	Tick *Waiter app and kitchen display* in the restaurant and make sure the user belongs to the same company.
Tickets do not appear on the KDS	Check the station of the dish (chapter 10) and that the screen shows the right station or Expo; look at the connection chip.
Everything works but nothing live	The websocket is blocked: route `/websocket` to the gevent port in your proxy (chapter 21) or set `proxy_mode` correctly; the app polls meanwhile.
Open a point-of-sale session at payment	Open a session for the restaurant in Point of Sale.
Ingredients are not consumed	The dish needs a bill of materials and *Consume ingredients* must not be *Never*; check the stock location.
The stock of the dish goes down twice	The dish is a stored product: make it not tracked in stock.
Arabic receipt prints as boxes	Set the print mode to *Image*.
Printer does not connect on iPhone	iOS supports only Bluetooth Low Energy printers; use a network printer or an Android device.
The app cannot reach a http server on iOS or Android	Enable local network access with the setup script option or use https.
A table stays red or blue	An order is still open: open Restaurant → Open tables, check it, or cancel it as a manager.

28. Support

> **In plain words:** Updates are free. Installation help and bug fixes are included for 6 months.

Installation help and bug fixes are included for 6 months, with lifetime updates. Contact us through the marketplace page where you bought the module or write to **support@taqnix.com** with your Odoo version and a screenshot.

Changelog

1.0.0 – Initial release

- Odoo module: floors and tables of the restaurant POS with live states and a drag-and-drop floor map, orders with seats, courses (hold and fire), forced and paid modifiers, kitchen notes, stations with smart routing, tickets with SLA timers, bump, recall, all-day count and expo view, transfer and merge of tables, sold-out (86), bills (even, item, amount splits), cash, card and QR payment, tips, invoices, POS orders in the open session, ingredient consumption from the bill of materials (multi-level) with waste tracking, reports
- REST API with idempotent operations and an event log, real-time events on the Odoo websocket, immutable local sequence numbers
- Flutter source of one app with Waiter and KDS modes, offline outbox, LAN and Bluetooth ESC/POS printing, English and Arabic

© Taqnix. Licensed under OPL-1.